

For Reference

NOT TO BE TAKEN FROM THIS ROOM

Ex libris
UNIVERSITATIS
ALBERTAENSIS



THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR Kathryn Ward
TITLE OF THESIS Measurement, Characterization, and
 Modelling of an MTS Workload
DEGREE FOR WHICH THESIS WAS PRESENTED Master of Science
YEAR THIS DEGREE GRANTED 1979

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

THE UNIVERSITY OF ALBERTA

Measurement, Characterization, and Modelling of an MTS
Workload

by



Kathryn Ward

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE

COMPUTING SCIENCE

EDMONTON, ALBERTA

FALL, 1979

THE UNIVERSITY OF ALBERTA
FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled Measurement, Characterization, and Modelling of an MTS Workload submitted by Kathryn Ward in partial fulfilment of the requirements for the degree of Master of Science.

Abstract

A technique, for characterizing an MTS production workload at the University of Alberta, is developed that can be used to provide a detailed workload model. The workload model is designed to be used as input to a general purpose system model. In addition it can be used to generate a synthetic jobstream.

The workload model expands upon previous studies done in the area of modelling workloads. A general methodology is developed for the measuring, characterizing, and modelling of a one to three hour production workload. The measurement is accomplished via a series of system snapshots which are grouped into a number of segments composing the measurement period. The data collected is fully validated before the characterization phase begins. Within each segment every task's consumption rate of each hardware resource is calculated and all such task profiles are then used to determine task groupings or clusters of these profiles. The resource consumption rates of each profile cluster are the averages of all the resource consumption rates of the member tasks.

The final model, built from the above characterization, specifies the characteristics of each task group within each segment, as well as specifics on the arrival and departure of each task within the group.

Acknowledgement

I wish to thank the Department of Computing Services, who made this research possible, by providing the computing resources. I also wish to acknowledge the support and encouragement given me by my colleagues within the Systems Group.

In addition I thank Professor Ursula Maydell, my supervisor, for her advice, criticism, and guidance during the research and preparation of this thesis.

Table of Contents

Chapter	Page
1. Introduction.....	1
2. An Overview of Workload Models.....	6
2.1 Design Decisions.....	6
2.2 Natural Models.....	10
2.3 Artificial Models.....	11
2.3.1 Executable.....	12
2.3.2 Non-Executable.....	13
3. Environment.....	17
3.1 Hardware.....	17
3.2 Software.....	21
3.2.1 System Facilities.....	22
3.2.2 System Structure.....	22
3.3 Usage of MTS at the University of Alberta.....	29
4. Design and Application of the Workload Model.....	33
4.1 System Model.....	34
4.2 Synthetic Benchmark Jobstream.....	35
4.3 Workload Model.....	36
5. Experiment to Collect Workload Data.....	44
5.1 Workload Model Data Collection.....	44
5.1.1 Conversion Software.....	46
5.1.2 Data Collected.....	49
5.1.3 Monitor Software.....	51
5.2 Evolution of the Workload Model Data Collection....	54
5.2.1 Pilot Study I - Determination of Workload Variables.....	54
5.2.2 System Modifications.....	56

5.2.2.1	Changes to the MTS Job Program.....	57
5.2.2.2	Changes to Terminal and Network DSRs.....	59
5.2.3	Pilot Study II - Determination of Data Collection Interval.....	59
5.2.4	Pilot Study III - Determination of Intersnapshot Wait Time.....	61
6.	Conversion and Validation of Workload Data.....	65
6.1	Observed Workload Data.....	65
6.2	Execution of the Workload Monitor.....	68
6.2.1	Overhead of Workload Monitor.....	68
6.2.2	Analysis of Snapshot Exposure Times.....	73
6.3	Data Conversion.....	74
6.3.1	Measurement Error.....	77
6.3.2	Error Adjustments.....	79
6.3.3	Software Tools.....	88
6.3.3.1	DSR Record Reduction.....	88
6.3.3.2	Workload Conversion.....	89
6.4	Data Validation.....	90
6.4.1	Verification Software.....	91
6.4.2	Validation Results.....	92
6.5	Converted and Validated Data.....	96
7.	Construction of Workload Model.....	98
7.1	Determination of Task Groups.....	98
7.1.1	Clustering Algorithm.....	102
7.1.2	Algorithm Implementation.....	110
7.1.2.1	Choice of Variables.....	110
7.1.2.2	Choice of Initial Cluster Centroids.....	112

7.1.2.3 Clustering Software.....	114
7.1.3 Clustering Results.....	116
7.2 Workload Model.....	121
8. Summary, Conclusions, and Future Research.....	123
8.1 Workload Model.....	123
8.2 Future Applications.....	127
8.2.1 System Model.....	127
8.2.2 Synthetic Jobstream.....	128
8.3 Future Research.....	129
Bibliography.....	131
Appendix A - Experiment Schedule.....	148
Appendix E - Clustering Results Data.....	149

List of Tables

Table	Description	Page
1	Workload Monitor Overhead	72
2	Collection Error Results	93
3	Lifetime Categorization Results	94
4	Total Number of Tasks by Segment and Cluster	160
5	Active Tasks by Segment Cluster - Part I	161
6	Active Tasks by Segment and Cluster - Part II	162
7	Actual Unscaled Consumption Rates - Part I	163
8	Actual Unscaled Consumption Rates - Part II	164
9	Actual Unscaled Consumption Rates - Part III	165

List of Figures

Figure	Description	Page
1	Hardware Configuration of Channels 0 through 7	19
2	Hardware Configuration of Channels 8 through 15	20
3	System Structure	24
4	MTS Usage	31
5	Sequence of Data Collection Snapshots	47
6	Format of a Workload Record	53
7	Format of Resource Consumption Rate Vector	76
8	Structure of a Segment	83
9	Task Groups - May 24, 10 a.m. - Segment 1	118
10	Workload Model - Segment Format	122
11	Task Groups - May 24, 10 a.m. - Segment 1	150
12	Task Groups - May 24, 10 a.m. - Segment 2	151
13	Task Groups - May 24, 10 a.m. - Segment 3	152
14	Task Groups - May 24, 10 a.m. - Segment 10	153

List of Figures (cont'd)

Figure	Description	Page
15	Task Groups - May 24, 10 a.m. - Segment 20	154
16	Task Groups - June 7, 2 p.m. - Segment 1	155
17	Task Groups - June 22, noon - Segment 1	156
18	Task Groups - June 27, 1 a.m. - Segment 1	157
19	Task Groups - June 27, 2 p.m. - Segment 1	158
20	Task Groups - June 28, 8 p.m. - Segment 1	159

1. Introduction

Computer systems today are an integral part of the day to day activities of our society. As our dependence upon them increases, and the funds allocated to them also increase, the overall performance of these systems becomes more critical. Not only is it crucial that the system be capable of meeting the demands requested of it, but there is an increased awareness that like any other organizational function it is a cost centre in its own right and all decisions regarding acquisitions must be justified. It is from these needs that the study of computer system performance grows.

A computer system, usually referred to as just 'system', is composed of a number of hardware and software resources. The area of system performance is concerned with how these resources respond to the demands or requests placed upon them by the users of the system. Running a large computer system is an expensive proposition and many critical activities may depend upon its reliability and performance. It is not only important that a computer system be available during specified production hours but additionally that a user can rely on a certain level of service during these hours. How this service can be provided at the most reasonable cost is still a pertinent question today.

System performance itself is not a well defined term. Its intended meaning must usually be taken from the context

in which it appears. Svobodova (SVOB76) breaks system performance into two components:

1. the effectiveness of the system; and,
2. the efficiency of the system.

The degree to which the system meets the expectations of its users is its effectiveness while the efficiency of the system is how well equipped the system is internally to handle user requests in the most efficient manner. This breakdown by no means covers all system performance definitions that appear in the literature. For example, one very different definition of system performance is simply the ability of the system to perform correctly. The term system performance henceforth will be used only in connection with effectiveness and efficiency as described above.

Neither the effectiveness nor the efficiency of a system can be studied without carefully considering the actual demands being placed upon the system. The term system performance then relates to the ability of the computer system to respond efficiently and effectively to a given set of user requests.

System performance studies generally involve studying the effect of changes to the hardware or software resources, where system performance measurements can be obtained either on a real computer system or on a modelled computer system. A modelled computer system is a simulated or mathematical model of the real system. In either case, experimental

results must be reproducible if comparisons of performance are to be made. Input to the system, real or modelled, necessarily must include the workload the system is to process. The validity of the experimental results depends directly on the input integrity, including the workload characterization. If an experiment is to measure production performance then the workload representation must reflect the production workload. Therefore in order to reproduce the experiment the production workload must be reproducible. Without such a facility system performance measurements are reduced to only monitoring the system at a given time during production and cannot provide a measure of the performance changes due to hardware and software resource changes. Hence, measurements of a real live workload, namely the resource demands, must be grouped according to specific demand characteristics, 'characterization of workload', which then can be used as parameters in 'workload models'. A validated workload model representing the actual workload can then be considered to drive a real or modelled computer system just as the production workload drives a real system.

The complexity of deriving a validated workload model of the actual system is not to be underestimated. Early workload studies were done at a very general level. As the requirement for more detailed workload representations developed the difficulty of this task started to be understood more. A paraphrased quote was given by Ferrari (FERR72) in a paper of this latter period. It was

'Blessed is he who found his computer's workload. Let him ask no other blessedness.'

The severity of the problem is reflected by the frequent inclusion of Ferrari's quotation in many workload papers presented today. Not only does the original problem remain a difficult one, but the computer systems involved are becoming increasingly complex thus increasing the difficulty in characterizing the workload. Virtual memory systems and terminal oriented systems are two such enhancements. These extensions alone greatly expand the complexity of modelling representative workloads.

The research reported here is a study in producing such a validated workload model. The initial design is based largely on the work of Bard (BARD76a, BARD76b, BARD77, BARD78), who develops a workload characterization for VM/370 and uses a corresponding workload model as input to a system model. The system model accepts the workload model and hardware configuration parameters as input and produces performance measures as output.

The workload model derived in this thesis is intended as input to a similar model, built for the Michigan Terminal System. The model envisaged is somewhat more general than that used by Bard, and hence several extensions have been applied to his workload characterization technique. These extensions provide a more useful workload characterization, retaining more information while not significantly increasing the complexity of the model.

An overview of the types of workload models encountered in the literature is presented in Chapter 2, which in turn depends on the information in the annotated Bibliography. Background information on the particular computer system studied is outlined in Chapter 3. The motivation and design of the workload model, the design for collecting workload data, and the validation of the data are presented in detail in Chapters 4, 5, and 6 respectively. In Chapter 7 the workload data is characterized and the model parameters presented. A brief summary of results and guidelines for future research constitute the last chapter.

2. An Overview of Workload Models

The concept of a workload model encompasses a vast and varying range of models. The range is a reflection on the many purposes and situations for which workload models are essential.

The following review and analysis of workload models is based directly on the annotated Bibliography. The text is meant to be read in conjunction with the Bibliography as details of individual references have not been duplicated and some references in the Bibliography are not mentioned in the text.

2.1 Design Decisions

A workload model is designed in conjunction with its intended purpose. Some workload models are complete by themselves while others are designed specifically as input to a system model. A workload model is usually designed to aid in making decisions about the hardware and software resource changes of a system. This includes studies such as:

1. selecting a new computer system;
2. changes to the current software and hardware resources that will improve current performance; and,
3. determining the performance of the computer system under some projected future workload.

Ferrari (FERR78) lists the characteristics of workload models as representativeness, reproducibility, flexibility,

simplicity of construction, compactness, usage costs, system independence, and compatibility. The aspects of these characteristics as they relate to workload models follow.

- Representativeness:

If there is no requirement for the workload model to be representative of some natural workload the task is simple. However the desire is usually to produce a workload model of the current production workload or a variant thereof. Hence representativeness is often the most critical characteristic.

- Reproducibility:

Any type of comparative study requires the ability to reproduce the experiment. Note that all the typical studies suggested involve such comparisons.

- Flexibility:

The workload at most installations remains constant over an extremely short period, if in fact at all. Hence, a flexible model is desired that can be used to express a constantly changing workload.

- Simplicity of construction:

The ease and cost of collecting the workload data and formulating the associated model are facets of the construction.

- Compactness:

The volume of data representing the workload demands of the total user population is often immense. Hence, in modelling the workload demands some form of

reduction is required to make the model parameters more compact and thus more useful.

- Usage costs:

The cost of using a workload model is dependent upon the use for which the workload model is intended. For example, a benchmark stream requiring stand-alone time on several machines will probably be costlier than a software model using a workload model as input to a system model.

- System independence:

If the study is to involve different hardware or software configurations the need for system independence must be emphasized.

- Compatibility:

A workload model must be compatible with the purpose for which it was intended. For example a jobstream used to run a benchmark would be of no use to a simulation model driven on resource allocation events.

These characteristics must often be considered as compromises. A model that is simple to construct and inexpensive to use is probably a model that is more compact. Unfortunately this may mean the exclusion of detailed factors that have a very significant effect on system performance.

An additional characteristic that clearly distinguishes workload models is their frame of reference. This can vary from the machine instruction level (KNIG66), to the per user

resource demand level per transaction (BARD76a, BARD76b), or to the number of user written programs executed (AGRA77). Closely related to the frame of reference is the system independence of the workload. The frame of reference is one of the foremost issues in the study of workload characterization today. As the frame of reference becomes broader, or less dependent upon the hardware and software in use, the workload dependence on the system decreases. If user service requests are considered rather than actual resource requests a greater degree of independence can be maintained. Bard (BARD77) builds features into an otherwise system dependent model to make it portable between varying hardware configurations with the same IBM operating system. It appears however that a truly system independent workload may have to take on a form very different from those in current use. Attempts to do this include the study by Rozwadowski (ROZW73) who proposes that workload be equated to mechanical work, and the study by Hellerman (HELL72) who proposes that any workload be converted to the memory requirements for its table look-up implementation.

Calibration and validation of a workload model is a non-trivial task. A workload model producing the same results on the real system, or a suitable system model, as the real workload produces on the production system is no assurance that the workloads coincide. At best the conditional statement can be made that given the current hardware, software, and system parameters the workloads

coincide. A small change in the system parameters may make system performance very dependent upon a workload characteristic that is overlooked, or considered in insufficient detail, in the workload model.

The great variety of workload models described in the recent literature, and their varying characteristics do not lend themselves to any obvious classification scheme. However, Ferrari's (FERR78) scheme surpasses others. In the scheme the primary level of distinction is a natural workload model versus an artificial workload model, where a natural model is composed of jobs taken from the production workload and any other form of model is referred to as an artificial model.

2.2 Natural Models

A natural workload, as mentioned above, is a model composed of a series of actual jobs extracted from the production workload. Undoubtedly its greatest virtue is that of being representative, given that an accurate cross-section of jobs are extracted. Natural workload models are usually based on a selection of jobs taken from the production jobstream verbatim. The entire jobstream for a given time interval may be selected or there may be an attempt to choose a representative subset (SHOP70). In most installations it is impossible to extract jobs completely as this implies, for example, that all user files can be returned to the state that they were in before the job was

run during production.

Often natural workload models are a subset of the workload models labelled as 'benchmarks'. A benchmark is a series of jobs that represent the real workload and can be run on a specified machine. If the benchmark consists of jobs extracted from the production workload it is a natural workload.

2.3 Artificial Models

The construction of a synthetic or artificial model is based upon some measurement of the natural workload. However, the workload model constructed is not a subset of the production jobstream.

A workload model can consist of a stream of jobs that can be executed on a real system or a description of the workload that is not capable of being run on the system. A model that can be executed by the system is an executable model. Note that natural models are executable by definition.

Ferrari (FERR78) continues his classification by separating those models which consist of a series of jobs from those that represent the workload in some other form. These types of models are referred to as executable and non-executable, respectively.

2.3.1 Executable

The usefulness of an executable, synthetic jobstream lies in the representativeness of the synthetic job. At the base lies a highly parameterized synthetic job that can be made to consume any combination of system resources. Most synthetic jobs continue to be based upon that proposed by Bucholz (BUCH69). The calibration of the synthetic jobstream is critical. A well designed calibration approach is discussed by Sreenivasan (SREE74). Where it is relatively simple to create a synthetic jobstream representative of a batch environment, the external factors involved in duplicating a terminal jobstream makes calibration much more difficult. The frame of reference upon which the job is based is still a critical question when the portability of the jobstream is of concern (MORG73).

This type of model is often referred to as a benchmark, as explained earlier. The more common term attached to an artificial, executable model is a synthetic benchmark.

To run a benchmark jobstream on a system, particularly in the case of an interactive load, a facility is required to take the supplied jobstream and recreate it as if it was the production jobstream. Such a facility is called a driver. The driver is used to represent the user's interactive load by simulating typical think times and processing the supplied jobstream in some well defined fashion. Drivers can be constructed at the hardware (SALT70) or the software level (FOGE72). A software driver must be

used with caution as the load it itself imposes on the system may distort an otherwise valid workload description. A natural executable model also requires the use of a driver to recreate the environment that originally generated the natural jobstream.

2.3.2 Non-Executable

A non-executable model is usually intended as input to a larger model, which can be anything ranging from a very complicated analytic model (BARD77, BUZE78) to a manual interpretation used in conjunction with internal timings supplied by the manufacturer (KNIG66).

Artificial, non-executable models are used primarily as input to mathematical and simulated system models. The workload models can be of a variety of formats. Most workload models of this type can be best classified as one of the following:

- **Instruction Mix:**

An instruction mix is a set of machine level instructions such that the frequency of various instructions equals the observed or measured use during a production run. It is generally used in conjunction with internal timings. As long as the frequency tables truly reflect the actual workload this technique can be used to compare the relative performance of two machines, for a given workload. A set of standardized benchmarks can be decomposed in this fashion to obtain

the relative performance of a machine (HILL66, BUCK69). The acceptance of this technique has decreased in recent years as the complexity of operating systems has grown and introduced many factors in addition to internal timings that affect overall performance.

- Probabilistic:

A probabilistic model takes the resource demands as random variables and attempts to describe system performance in terms of these variables. Although this has been done using regression (ANDE72, WALD73) the independence of the resource demand variables cannot be relied upon and while a regression model may perform well for the majority of jobs it may be incapable of describing any outliers (GOMA76). The application of a time series model and a Markovian model are considered by Agrawala (AGRA76) who treats the workload as a stochastic process.

- Trace:

Data collected from the system, recording the precise order of low level events, is referred to as trace data. The literature abounds with trace driven system simulators (MEAD78, SCHA75, SHER72). The method can be very good in conjunction with a simulated system model, for the trace itself retains information on the decisions made by the system as to which job should receive what service. This greatly simplifies the internal requirements for the simulated system model.

Trace data tends to be voluminous and sheds very little light on the overall characteristics of the workload.

Its detailed nature makes it useful for other purposes. For example trace data is used by Winder (WIND73) to gain insight into program behaviour to assist in architectural design.

- Resource Request:

Workload models used in conjunction with analytic models are often based on the resource demands of the users (BARD77). In some ways this type of model may be considered to be a trace model. However, the literature is quite clearly split between those models that are based heavily upon the order of the requests versus simply a summary of the requests. Resource request models are those fitting the latter description.

Within the category of resource request models the measured resource demands can be further reduced by categorizing the users according to their demands. This may be done in a manner dependent upon the system at hand as by Bard (BARD76a, BARD76b) where they are grouped according to the response of the scheduler to their demands. Alternatively Chanson (CHAN77) appears to group users according to their functions. A more scientific approach is taken by Agrawala (AGRA76) and Fangmeyer (FANG76) where automatic clustering techniques are employed.

The selection of resources to be included within a

resource request model is the other important aspect of data reduction. Two different ways of doing this are described by Agrawala (AGRA77b) and Mamrak (MAMR77).

The model designed and implemented in this thesis is a resource request model which specifies the demands that groups of users make upon each hardware resource over time. The model retains time specific information by considering the rate of resource demands with regard to the periods over which these rates were in effect. In so doing the model attains a degree of preciseness usually only attached to trace models, while not having to deal with the massive amounts of data associated with a trace.

3. Environment

The particular system under consideration is the system operated by Computing Services at the University of Alberta. It is the main installation for reasearch and teaching, the administrative computing is performed on a separate system. The main CPU is an AMDAHL 470 V/7 and the operating system used is the Michigan Terminal System, more commonly called MTS. It is a virtual memory, time-sharing operating system that offers both batch and terminal facilities.

3.1 Hardware

As in many installations Computing Services is experiencing rapid growth in the demands being placed upon the computing facilities. Due to this growth the hardware is constantly being replaced, upgraded, and supplemented; hence, changes occur frequently.

The AMDAHL 470 V/7 mainframe has eight megabytes of main memory and sixteen channels. Attached to the channels is a great variety of equipment including:

- magnetic tape drives;
- disk drives;
- paging drums and a paging fixed head file;
- card readers;
- line printers;
- terminals; and,
- minicomputers that operate as front end communication

processors.

The precise configuration of the devices on channels 0 through 7 is given in Figure 1, and on channels 8 through 15 is given in Figure 2. While the purpose and use of most of the hardware components is self-explanatory the disk drives and minicomputers require further explanation.

The two types of disk drives, IBM 3330's and IBM 3350's, are used for several purposes. The IBM 3350 devices are used to support the file system provided by MTS. Here reside all user and system files as well as all of the file catalogue information. The IBM 3330 devices are used for two system operations. The spooling system uses two of the drives for spooling packs, and a third drive is used for a paging pack which is used to migrate pages to when the primary paging devices are full.

The 'front end communication processors', called the FECP's, provide several services. Primarily they control terminal access for all terminals other than the IBM 3270 type devices. The FECP's operate in conjunction with the 'private automatic computer exchange', more simply referred to as the PACX. The PACX is an electronic terminal exchange that connects incoming calls to the FECP's and passes data between the terminal and the FECP's. Facilities also exist for direct lines into the FECP's, that do not pass through the PACX, as well as support for dial-up lines. The FECP's themselves control the transferring of data between terminals and the main system. In addition the FECP

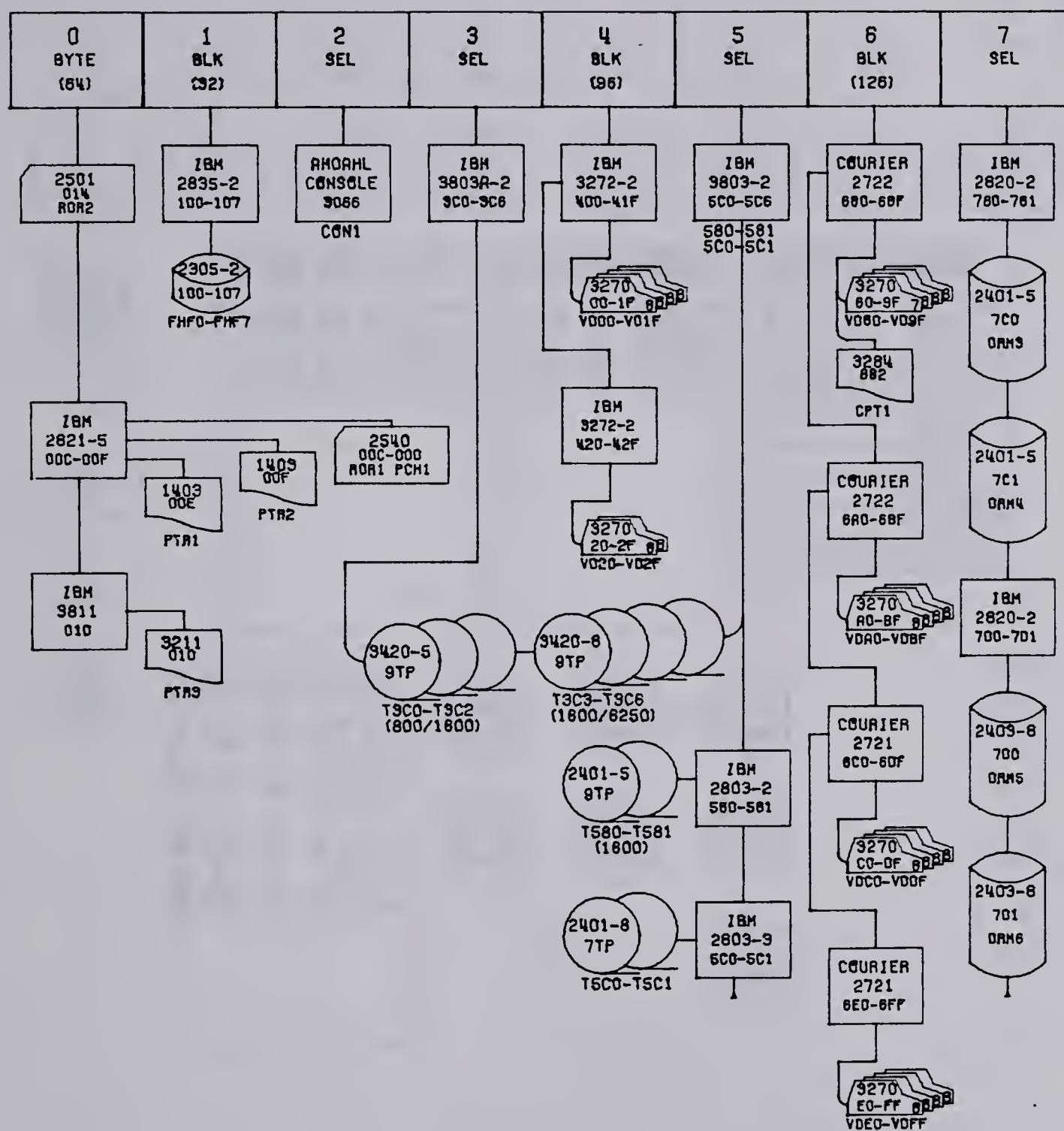


FIGURE 1 - Hardware Configuration of Channels 0 through 7

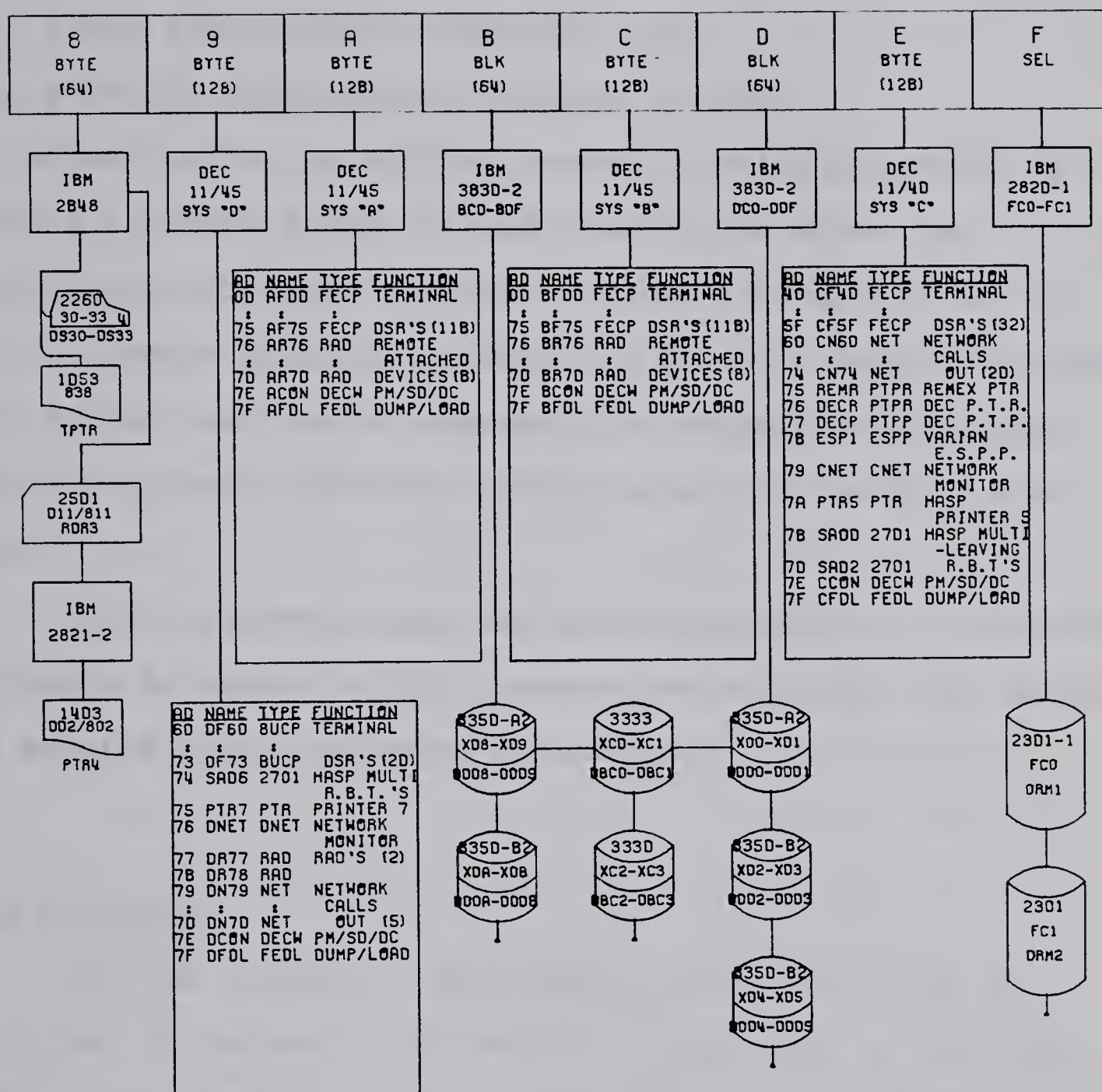


FIGURE 2 - Hardware Configuration of Channels 8 through 15

equipment allows access of a variety of peripheral equipment. These peripheral devices include:

- two paper tape readers and one paper tape punch;
- three remote batch terminals; and,
- a Varian electrostatic printer plotter.

A connection to the DATAPAC network is also supported by the FECP's allowing access to and from remote sites. An additional facility, supported by the FECP's, is that of RAD'S (remote attached deviceS). A RAD is a non-interactive I/O device that can be mounted, for example such devices include cassette recorders and printers of the DEC LA180 type.

At the present time, MTS at the University of Alberta supports in excess of 500 interactive terminals. The number of terminals is increasing continually.

3.2 Software

MTS was originally developed by the University of Michigan to run on an IBM 360/67. It now runs on both the IBM 360 and IBM 370 product lines as well as compatible mainframes such as the AMDAHL 470. MTS is run at eight installations in four countries; U.S.A., Canada, England, and Brazil.

Although each installation runs with specific modifications to meet their unique needs the system is continually expanding due to the development effort contributed by all installations.

As the name implies, MTS is designed primarily for access via interactive terminals, however the same computing facilities are also available to batch users.

3.2.1 System Facilities

MTS is an extremely 'usable' system that provides a wide range of services for every level of user. Among the general system facilities there exist:

- a general file system, supported by a file editor, that allows for detailed permission access of each file to specific users or programs;
- a powerful, but easy to use, command language through which all system requests are made;
- a symbolic debugging system; and,
- a variety of language processors covering procedure oriented languages, assemblers, interactive languages, list and string processing languages, simulation languages, application packages, and graphics.

Each potential user of the system must register to acquire a Computing Services id (CSid). Access to the system is gained only through a registered CSid. Associated with each CSid is a secret password. A session begins only after a valid CSid and its associated password have been given.

3.2.2 System Structure

It is not the intent to describe the internal structure of MTS in great detail. However, if the workload

characterization is to be fully understood a certain degree of background is required.

At the core of the system is UMMPS (the University of Michigan multiprogramming system) more commonly referred to as the supervisor. Its purpose is twofold:

1. to provide an interface between MTS and the hardware;
and,
2. to allocate system resources such as I/O, CPU time, and memory to the tasks.

For the purpose of this discussion the supervisor executes in system state while all other execution occurs in problem state. Supervisor state is that state in which all instructions are valid, including privileged instructions that are not available in problem state.

Above the supervisor several tasks are running at any one time. Each task, or job, is of a particular type depending upon which job program it is running. The internal system structure is given in Figure 3. All users run the MTS job program and hence are referred to as MTS tasks. The other job programs provide a variety of system services. One job program is the PDP (paging drum processor) which performs the actual reading and writing of pages to the paging devices (the supervisor decides which pages are to be read or written). Another job program is HASP (the Houston automatic spooling priority system) revised to run under MTS, which provides the spooling functions. Whereas several MTS tasks can be active at any time, only one PDP or HASP

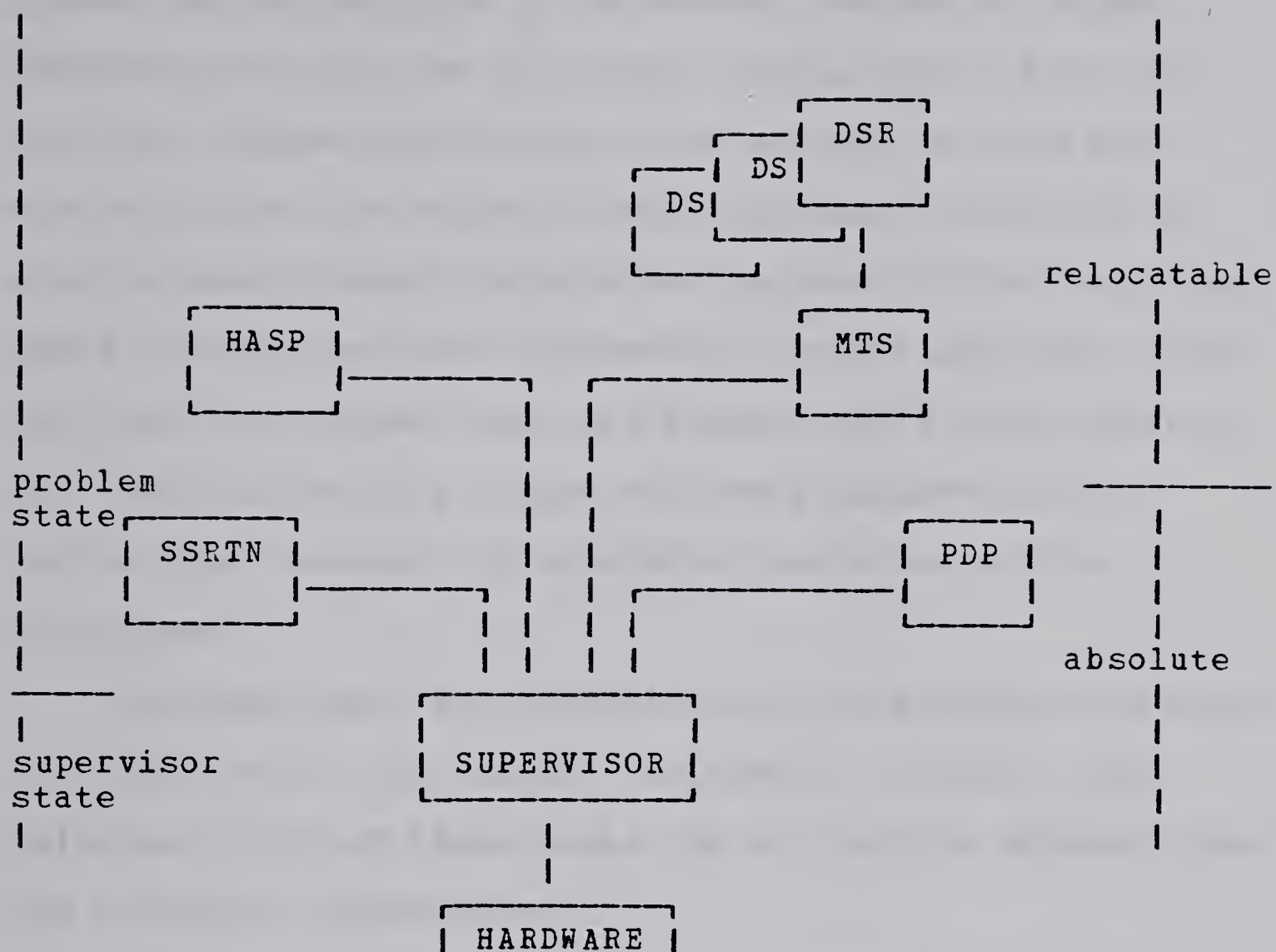


FIGURE 3 - System Structure

task is active at a time.

One further job program to be considered is SSRTN (the system status routine). It maintains a vector of values describing the load on the system. These load values are used for various functions, as for example to determine whether or not the number of batch streams allowed to run simultaneously should be modified. By use of this technique, SSRTN somewhat controls the external scheduling, the batch jobs that are allowed into the system. SSRTN also controls the modification of a number of system parameters that control the internal CPU scheduling performed by the supervisor.

Although there are several other job programs the above will suffice for the current explanation. Figure 3 also indicates which of these tasks run in absolute address space and which are relocatable.

Furthermore, the DSR's (device support routines), associated with MTS tasks are also illustrated in Figure 3. For each MTS I/O device there exists a DSR. The DSR constructs the actual I/O requests which are eventually processed by the supervisor and the specific I/O device. The actual structure of the DSR is dependent upon the device.

Each task, regardless of type, has a job table allocated to it by the supervisor. The supervisor maintains a fixed number of these job tables and allocates a free one to each new task. Each job table is a fixed length area where variable information pertaining to the job is stored.

For example the task's registers are stored in the job table when the task is not in execution.

To understand some portions of this study it is necessary to take a look at the CPU scheduling in a broad sense. There is a single main CPU queue which may contain tasks that are in various processing stages. Whenever the CPU is released the first dispatchable task closest to the top of the CPU queue is permitted to use the CPU. The various processing stages of a task are:

1. running, being processed by the CPU;
2. ready, it could use the processor if it were made available to the task;
3. waiting on some event, until some interrupt occurs or until some bit in memory is reset; or
4. in page wait.

The main CPU queue contains:

1. all tasks that are ready; and,
2. all tasks that are waiting for some event of which they will not be automatically notified.

A task is removed from the main CPU queue when:

1. a wait is initiated by the supervisor for either an I/O operation or a page read;
2. a wait is initiated on a bit in memory that the user will be notified of; or
3. the task requests to be removed.

When a task is returned to the CPU queue it is placed at the top, such that it is next in line for the processor. This

provides quick service for interrupts and tasks that have just had a page read in. To overcome the problem of several large jobs monopolizing the top of the CPU queue and creating a thrashing situation a 'privileged' job mechanism exists.

When a task is initially added to the CPU queue it is allocated a time slice (an amount of CPU time) and a paging slice (a number of page reads) and then becomes neutral. As the task is processed, it is moved off of the CPU queue and back onto the CPU queue, its time and paging slice being decremented as they are used. Each time it is returned to the CPU queue it is placed at the top of the queue but does not get a new paging or time slice. Eventually, if the task is not completely processed it will:

1. exceed its time slice;
2. exceed its paging slice; or
3. request a new slice.

This event will be referred to as slice end, regardless which of (1), (2), or (3) occurs. At this time the task is allocated a new time slice and new paging slice and is placed at the bottom of the CPU queue.

The supervisor calculates an estimate of the number of real pages the job requires, or its working set size. The supervisor's estimate of the working set size is calculated at slice end as follows. If slice end is caused by case (1) or (3) then the working set size estimate is set to the number of real pages it had in real memory. Otherwise, if

caused by (2), the new estimate is the number of real pages it had plus one half of the number of virtual pages that it currently owns which were not in real memory. This estimated working set size is subject to four additional constraints:

1. The estimated value cannot exceed more than the current number of real pages plus a constant.
2. The estimated value cannot exceed more than the number of virtual pages owned by the task.
3. The estimated value cannot exceed more than an internal limit set within the system.
4. If slice end was caused by case (2) then the estimated value cannot be less than the previous estimate of the working set size.

Whenever a task is dispatched the estimated working set size is checked against a threshold value for a 'big job'. A big job which is dispatchable is called a 'privileged' task while one which is not dispatchable is called a 'non-privileged' task. The term privileged is a bit of a misnomer as in no way does a privileged task gain rights that other tasks may not have. All it does obtain is an extended time and paging slice. If the working set size of this task plus all other privileged jobs is less than a certain threshold then the task becomes privileged, otherwise it becomes non-privileged. When a privileged task reaching the end of its slice has more pages than the big job threshold or exceeds its allotted page reads, it becomes non-privileged otherwise it becomes neutral. In either case

it is returned to the bottom of the CPU queue. Its removal from the set of privileged jobs reduces the number of pages allocated to privileged tasks and the scheduler locates a non-privileged task whose working set size will not cause the number of pages in use by privileged tasks to exceed the maximum limit. This task then becomes privileged. Some of these threshold values are dynamically modified by SSRTN based on the load on the system.

3.3 Usage of MTS at the University of Alberta

The users of the system are composed of the University community, staff and students, as well as a number of external clients who are classified according to educational, non-profit, and commercial. The bulk of the load comes from the University users.

All resources consumed are charged for on the basis of a sliding scale where University users pay the least and commercial users the most. Costs are lower during evenings and weekends. At the termination of each MTS session the user is notified of the resources consumed and the total cost. This data is additionally recorded in a statistics record for that session. At month end all statistics records generated are processed and the actual billing performed. The resources charged for and hence recorded are:

- CPU time;
- virtual memory used integrated over CPU time used;
- magnetic tape connect;

- paper tape connect;
- paper tape punched;
- cards punched;
- cards read;
- pages and lines printed on line printer;
- disk file space;
- plotting paper and time;
- electrostatic printer plotter paper and time;
- network time and data transferred; and,
- terminal connect time.

There are in fact three types of MTS tasks:

1. terminal;
2. batch; and,
3. system.

Two types of system tasks exist. The first is a task that is started up by the operator and has the script for its session prepared in a file. The second is an MTS task run from the operator console. Although they can be distinguished, for the purposes here it is adequate to consider them together.

The magnitude of the load on the system is highly dependent upon the period of time in which it is measured. This fact in addition to the rapid growth in demand experienced in recent years is illustrated in Figure 4. Each 'run', in the Figure, is either a batch, terminal, or system session as recorded by the creation of one statistics record, where a session is the use of an MTS task by one

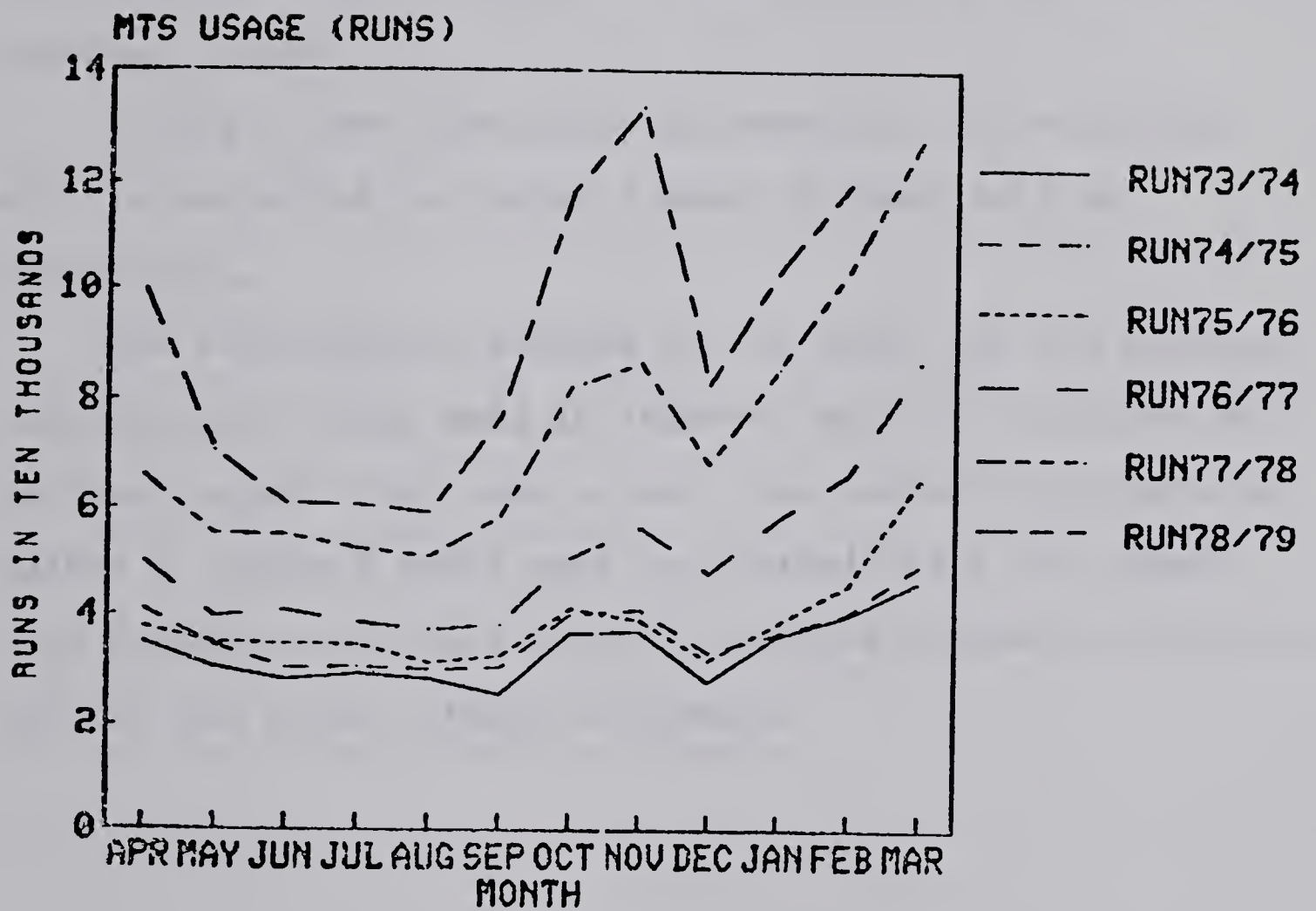


FIGURE 4 - MTS Usage

user.

Terminal access accounts for the majority of sessions and all indications point to more terminal usage and less batch activity. Even the demand on the traditional student batch facility, where students are assured quick turn around on small jobs, is decreasing as the increased availability of terminals makes it possible for courses to depend upon terminal access.

During a busy afternoon, in excess of 200 terminals will be connected and about 5 batch streams will be operational.

The everchanging demands of the users and the hardware modifications being made in response make the creation of a workload model fixed over a long time period inappropriate. Rather a workload model must be created which can accept time dependent workload characterization parameters and will reflect the actual change in demand.

4. Design and Application of the Workload Model

System models can be used to aid in software and hardware resource decisions. A very valuable tool would be a system model that could be used to:

- tune the current software resources to improve performance;
- study the effect that the change in a particular hardware or software resource would have on system performance; and,
- determine the performance of the current system with a projected workload.

The long term goal is to produce a system model of MTS that would allow these types of studies to be done.

A workload model is required to supply input to such a system model. In this study such a workload model is designed and constructed. Since the workload model, its structure and variables, is dependent on the type of queries which are to be answered about the system it is necessary to at least conceptually describe the system model, as presented in Section 4.1. As mentioned in Chapter 2, Ferrari (FERR78) also emphasizes the close link of the workload design and its ultimate application. An analysis of the type of workload model required is presented in Section 4.3. Furthermore, in designing the present workload model it became evident that it also would lend itself to the construction of a synthetic jobstream, a brief description of which is given in Section 4.2. The existence of such a

jobstream can be used for tuning the current software resources. However, it is emphasized again that the present study concentrates on the workload model and that detailed studies of a synthetic jobstream and system model are beyond the scope of this thesis.

These two applications are presented and then followed by an analysis of the type of workload model required.

4.1 System Model

A system model that can be used for all software and hardware resource decisions is an extremely powerful and complex tool. It must be precise and specific so that it can be used to study a change in any area. Additionally, due to the continual change in the hardware and software resources in question, it must be designed to be flexible and easily modified. As mentioned earlier a model similar to this is presented by Bard (BARD76a, BARD76b, BARD77, BARD78). His model allows for hardware parameters and a workload model as input to determine the effectiveness of VM/370 given the supplied hardware configuration and workload demands. The system model proposed for MTS is somewhat more general, allowing for studies of software resource changes as well.

With the above concerns in mind, the implementation is envisaged as a modular model. Each module would be identifiable as modelling a certain software resource. Such a structure ensures that a well defined change to the real system could be quickly identified and implemented within

the system model. The model would be hybrid, composed of mathematical as well as simulated modules, as discussed by Svobodova (SVOB76). A particular application of this type is described in the paper by Gomaa (GOMA76) who employs simulation and regression. In the current study the mathematical modules are envisaged as using queuing techniques, such as used by Bard, and the model would resort to simulation techniques in areas where mathematical techniques are found to be unsatisfactory.

Software, hardware, and workload parameters would form the input to the model. As little as possible software dependent information, as well as hardware resource dependent data, would be built into the actual system model. The basic software structure, such as that illustrated in Figure 3, would be built into the model in such a fashion that each system function could be clearly identified for modification. A number of the software resource parameters would be supplied as input but the basic software resource structure would be built into the system model itself. As mentioned before a model of this complexity is highly dependent upon valid workload model parameters.

4.2 Synthetic Benchmark Jobstream

The type of workload model that drives a system model, such as that described above, also lends itself to the creation of a synthetic jobstream. It is, however, a much less complex and not nearly as powerful application as the

system model, but nevertheless is a useful facility.

The resource consumption data that is required to drive the system model can also be used to derive a synthetic jobstream that will make the same resource demands on the computer system. This ability to recreate a production jobstream is useful for tuning as well as for benchmarking.

4.3 Workload Model

The basic requirement of the workload model is to account for all consumption of the hardware resources. Such a broad requirement can be approached in many different ways. The actual technique chosen must be developed in conjunction with the required features of the workload model. Included in these features are: the resources to be measured; the frame of reference of the resource measurements; detail of the data within the workload model; the real time period that the workload model represents; the time frame which each measurement represents; and, the consideration of overhead activities.

- The resources to be measured:

The model should represent all hardware resource consumption that occurred. The workload characterization by Bard, as well as the study presented here try to accomplish this.

- The frame of reference of the resource measurements:

The frame of reference, or units of resource requests, for the workload model is critical. For example, when a

user issues an MTS command, it will clearly make demands on several hardware resources, and is subsequently processed down through numerous processing stages of the operating system terminating in hardware demands. At which stage should the request be recorded? The workload model itself is to form input to a system model that reconstructs the software resources of the system. Therefore resource requests cannot be considered at the hardware or machine level for it is the purpose of the system model to accept the request and process it down to this lowest level. Alternatively too high a processing stage makes it impossible to collect sufficiently detailed resource request data. An example is the use of disk files. Collected at the command level the information retrieved may simply be the number of times a user copied one file to another. However, this would not include information on the amount of data actually copied. At the other extreme the information collected could be the number of start I/O instructions issued to the disks. This however, is dependent upon the file system software and would make it very difficult to test the performance of a new file system, the input data being dependent on the old file system. Elaborating on this suppose that a request to read one given line from a particular file is subsequently accomplished by one or more start I/O's to the disks as generated by the file system. The number of lines read from and written

to the file routines, in addition to the number of times a file was opened, gives a specific measure of file system activity without dependence on the structure of the file system. If the file system module of the model were to be changed these values, number of lines, would still be meaningful as input since the I/O activity is measured above the file routines. However these I/O measures would not be valid if the entire command structure of MTS changed since it is the command structure that is creating the individual reads and writes. Therefore it is the number of initializations of the device, in addition to the number of lines read from and written to it that are measured.

This is a different approach from that used by Bard, who studied the lowest level hardware requests available. For his study the software resources are considered as fixed so that such a frame of reference for the resource units is valid.

- The detail of the data within the workload model:

The detail of the data is closely related to the volume of data. If and how the data can be reduced is an important question, for example whether or not each task should be considered individually in the finalized workload model. This may result in a workload model that includes several hundred workload profiles. Any benefits gained from considering such a detailed design would probably not compensate for the complexity and overhead

involved in a system model constructed to handle such a workload model. Additionally, generating a synthetic jobstream from such a workload model would be impossible. Therefore the approach is to reduce the hundreds of task profiles to a smaller number of well defined task groups. Each member of such a task group will have a similar workload profile, based on the average resource consumption of all tasks in the group.

While both Bard's study as well as the current study impose such a classification on the profiles, manual classification is used by Bard whereas automatic clustering techniques are used in the current study to let the profiles group naturally. This technique ensures a more general facility in classifying large numbers of tasks.

- The real time period that the workload model represents: A workload model can represent the consumption over a few minutes, a few hours, or even a number of days. When designed for use with a system model the period chosen must be short enough to allow the system model to process the workload in a reasonable time but long enough to provide a representative workload with adequate time to allow the system to react to the load placed upon it. Because averages are being used to represent a given user class, too long a period may tend to lose significant distribution information within the data. For example a very heavy load in one half hour

period followed by a very low load in the subsequent half hour is not adequately represented by stipulating an average load for an hour's duration. For this reason the present study considers a total period of one to three hours containing subperiods, called segments, to which the averaging techniques are confined.

Bard considers the same total length of period but averages the results across the total time period. Hence, the current study provides a better representation of the variation in the load over the total period.

- The time frame of each measurement:

The model is built upon resource consumption rates rather than the more usual technique of considering actual resource consumption. A simple example shows the importance of this approach. Suppose each of two jobs consumes M units of resource R in a given period. Let job 1 be a task that was observed to consume this resource over a period of t minutes, and let job 2 be a task that was observed to generate the same total request but over a period of $10(t)$ minutes. Job 1 placed heavier demands on the system resource R over a shorter period of time while job 2 placed lesser demands over a longer period. Their impact on the system over any chosen interval is quite different. In terms of consumption rates, the two jobs have very different resource consumption data and hence would belong to

different task groups which in turn more clearly describe the instantaneous load they place on the system. Using the measure of total resource consumption the jobs would both belong to the same task group, which in essence discards the information about the distribution of the resource demands. That workload model would then have to be equipped with the ability to distribute these total requests over given time periods. In fact it would be attempting to rebuild distribution information that had been thrown away.

Both Bard and the current study use rates in attempts to overcome the information loss incurred by the total resource demand approach. Bard's consumption rates are per transaction where a task can consist of hundreds of transactions. A transaction, typed as trivial or nontrivial, is the unit of work passed to the scheduler. Users are grouped on the basis of their transaction types. The current study bases its rates on usage per user over real time. The largest real time period is a segment; however if the task's lifetime is less than a segment then the lifetime period is used.

- The consideration of overhead activities:

While a user requests specific resources directly of the computer system the operating system software may allocate additional resources. For example in a paging system the user simply requests main memory space, yet in being allocated this memory space the user is

additionally allocated space on a paging device as well as the resources necessary to move the pages to and from the external paging device. Although these additional resources can be defined as overhead due to the software behaviour, the actual resources allocated are completely dependent upon the external demands being placed upon the system at the time the original request is made. If a workload model is constructed that is used in conjunction with a real or simulated system and succeeds in reproducing the requirements of primary resources yet fails in reproducing the overhead resources then the model has failed. The ability to include and exclude particular tasks from the workload model must therefore be kept flexible such that the direct user requests and the corresponding overhead demands can be dealt with independently once the system model is fully designed and implemented.

These aspects of workload modelling cover a wide spectrum of options and modelling techniques. The actual choices made were not all obvious from the onset but developed through experimentation and subsequent analysis. Although Bard provided the initial design many aspects were expanded to provide a more usable model as noted above. In summary the major generalizations are:

1. breaking the required observation interval into segments to prevent loss of peak data;
2. using consumption rates over sequential time periods

rather than basing the rates on internal scheduler characteristics;

3. measuring the resource consumption in such a way that enough detail is available for a system model intended to study changes to the software resources in addition to changes to the hardware resources; and,
4. replacing manual grouping of tasks by an automatic technique.

5. Experiment to Collect Workload Data

The development of the tools to allow a valid set of workload data to be collected was an involved process. The tools to monitor the workload evolved in conjunction with the installation of data collection points within the system that provided necessary data. The required data collection points and the desirable properties of the monitor became clear over a series of experiments. The final result is presented first and followed by the history of its development.

5.1 Workload Model Data Collection

The data collection mechanism consists of an MTS task that is set to take 'snapshots' of system activity at predetermined times and intervals, a snapshot records the current status of the resource consumption of all active tasks. This MTS task is referred to as the workload monitor, or simply the monitor. One invocation of the workload monitor provides a collection session, constituted of 640 snapshots. Upon completion of each snapshot the monitor goes into a 5 second real time wait and then commences the next snapshot. The 640 snapshots are grouped into 20 segments of 32 snapshots each.

The monitor operates on a per task basis. All active tasks are found by scanning for all active job table entries. Each active task encountered by the monitor during

a segment causes a data record to be set up to retain the data collected about it. The data record for a task contains detailed information on the resources it consumed during any of the segments for which it was active. The data records are maintained in virtual memory. A data record consists of a header section and from one to twenty data sections. When a new task is found a header section is set up for it, and its current consumption of all resources is recorded as origin values along with the task identification information. A data section for the current segment is also established and linked to the header section of the record. On all subsequent snapshots, during this segment, the current consumption of all resources of this task are recorded in the data section. At segment end an additional data section is linked onto the data record of each active task and is used to retain consumption data for the next segment.

On each snapshot the job table list is scanned, all inactive table entries being ignored. An active task causes the active list of records in virtual memory to be searched for a match on task number and in the case of MTS tasks on the CSid as well. If a match is found the current data is updated to current status. If no such record exists a new record is constructed and joined into the active list. The initial data and identification are set to the current status.

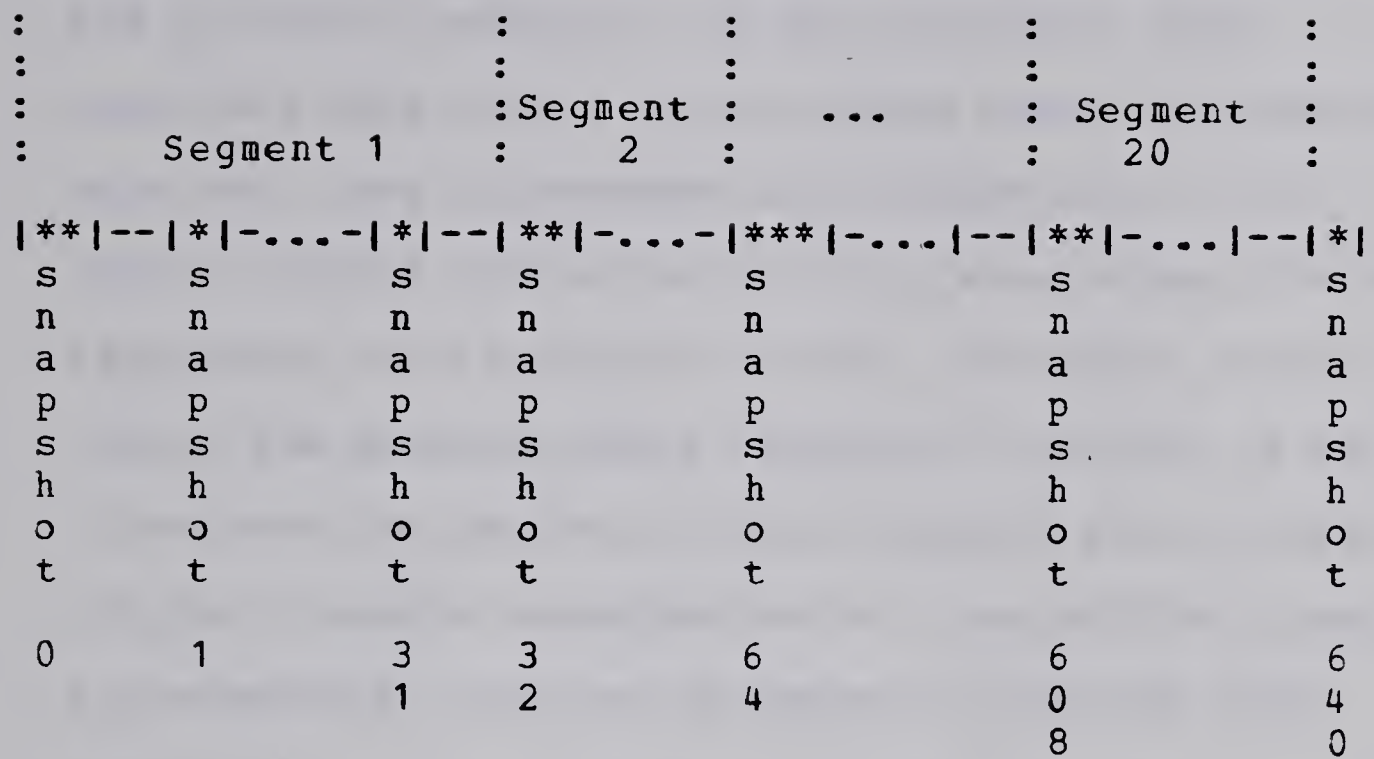
Each data record is written out to a disk file when

either the task is completed and is no longer active or when the 640th snapshot is completed, whichever occurs first. The structure of this collection mechanism is shown in Figure 5. For the purpose of analysis a definite split between the end of one segment and the beginning of the next segment must be determined. The first segment ends at the start of snapshot 32, and the second segment begins there. Similarly the second segment terminates at the beginning of snapshot 64 and the third segment begins there. The entire session, composed of the 640 snapshots, is called the collection interval or session.

5.1.1 Conversion Software

The data used for collection and validation comes from three sources:

1. For each active task the system retains within its internal data structures specifics of the resources it is consuming. Some of these values are maintained for accounting purposes, some for the system scheduler, and others simply for the use of monitoring activities. This data can only be captured by means of a separate task collecting and storing it. The collected data comes from two data structures. The first is the job table which exists for all tasks regardless of type. The second is the MTS system area which exists only for MTS tasks. The data found in the second category is data relating to I/O activity and response times.



** snapshot time
 -- intersnapshot wait time

FIGURE 5 - Sequence of Data Collection Snapshots

2. The device support routines, DSR's, for terminals and the DATAPAC network can be directed to record information on the number of input and output operations and the total length of the data involved. This recording only occurs when a system switch has been set. Once set, data is recorded and written out to the general system collection facility when either the task terminates or the switch is reset, whichever occurs first. The general system collection facility is used throughout the system to store required data records. All data records deposited are of prespecified types and are eventually retained on magnetic tape for later retrieval.
3. During normal production the load information retained by SSRTN is recorded every twenty seconds. This gives the accumulated load placed on various resources over the last twenty seconds. These records are referred to as loadlevel records. This loadlevel data is used entirely for validation of the workload data obtained from sources (1) and (2). The loadlevel records are recorded via the general system collection facility and hence are available for processing at a later time. From the loadlevel records the accumulated idle time and accumulated number of page I/O's are used in conjunction with the times the measurements were made to calculate the actual CPU time used and the number of page I/O's performed.

5.1.2 Data Collected

The variables collected from the internal data structures are classified as stage I and stage II, where the stage I data was available from the beginning while the stage II variables became available only after system modifications were made. The variables observed are:

- task identification - (stage I- from job table):
 task number
 task type (i.e. MTS, PDP, or HASP)
 for MTS type tasks the CSid and the MTS job type (i.e. terminal, batch, or system);
- supervisor state CPU - (stage I - from job table);
- problem state CPU - (stage I - from job table);
- page I/O's - (stage I - from job table);
- printer pages printed - (stage I - from MTS system area);
- virtual memory size - (stage I - from job table);
- supervisor's estimate of working set size - (stage I - from job table);
- unit record activity - (stage II - from MTS system area);
- magnetic tape activity - (stage II - from MTS system area);
- disk file activity - (stage II - from MTS system area);
- terminal activity - (stage II - from MTS system area)
 (also available as stage I from DSR system area but improved at stage II);

- electrostatic printer plotter activity - (stage II - from MTS system area);
- paper tape activity - (stage II - from MTS system area);
- RAD activity - (stage II - from MTS system area); and,
- network activity - (stage II - from MTS system area).

The last 8 variables compose the MTS I/O counts, measures of the I/O activity of MTS tasks to these devices. The activity data includes the number of initializations of the device, the number of inputs with the total length of data from all inputs, and the number of outputs with the total length of data from all outputs. The precise meaning of initializations is device dependent, for example for magnetic tape it is the number of tape mounts while for the file system it is the number of times a file was opened.

The data from the DSR collection facility was used to correct values monitored at the MTS level. Both network users and terminal users have available to them a variety of device commands that control the behaviour of the device and allow special functions. These device commands are not processed by MTS but rather by the DSR alone. These requests can cause additional I/O to be done. Because the DSR alone sees and carries out the request, MTS does not add the I/O caused by these requests into the totals maintained in the MTS system area. The terminal and network DSR's therefore additionally count the I/O completed, the I/O visible to MTS and the other.

Some resource consumption that non-MTS tasks are doing

is lost. In particular I/O activity that does not pass through the MTS job program is not recorded. This is not a critical problem for in most cases the non-MTS tasks represent system overhead which will be removed from the workload representation since that is part of the operating system activity to be incorporated into the system model.

The workload monitor is activated based on an internal table of times and dates. The table is established such that data for nearly every day and hour combination will be obtained over a series of weeks, yet the monitor will go inactive for at least 2 hours between any two collection sessions.

5.1.3 Monitor Software

The monitor facility is composed of two assembler programs, a driver and a data collector. The driver program is started as part of the IPL (initial program load) process. It has an internal table which gives the times that the collector job should be started. Once started, the driver finds the current date in the table and then the next time at which a collection should be started. It then goes into wait state until that time.

At the end of the wait a file is created with a name of the form QQmddhh, where m is the last digit of the current month, dd is the current day, and hh is the start up time based on a 24 hour clock. These files are referred to as QQ files. The collector program is then called and generates a

collection interval during which the observed data, as described above, are written to the QQ file. When the collector terminates, control returns to the driver which once more calculates a real time wait to extend to the next collection interval. Furthermore, the driver, before calling the collector program, sets a system switch which triggers the terminal and network DSR's to start generating records. This switch is reset at the end of the collector program .

If the last collection interval of the day terminates prior to midnight then the driver program starts up another MTS task to clean up the data files. All QQ files are copied to two tapes and then destroyed. In addition the file to which the DSR statistics are deposited is copied to tape and emptied.

The collector program performs all data collection as described above. Each task observed has a record in the QQ file. The record consists of the initial data as well as identification. For each task there also appear up to 20 data sections, one for each segment during which the task was active. The record is written out when the task is first observed to be missing or on termination of the last snapshot. The data retained is given in Figure 6.

At the end of the snapshot all tasks whose records were not updated on the current scan are written to the QQ file. At the end of the segment each active record is extended to include a data section for the subsequent segment.

One additional record written to the QQ file giving the


```
*****
***** Identification and Origin Values *****
All accumulated values are up to time of first snapshot on
which task is observed.
```

- UMMPS task number.
- Task type (i.e. PDP, HASP, MTS, etc.).
- If MTS task then CSid and type: terminal, batch, or system.
- Snapshot first observed and time of observation.
- Snapshot first observed missing and time of observation.
- Accumulated problem state CPU time.
- Accumulated supervisor state CPU time.
- Accumulated page I/O's.
- Accumulated pages printed.
- Accumulated MTS I/O counts.
- Accumulated number of responses and total response time.
- Accumulated number of think periods and total think time.

```
*****
```

```
*****
*** Data section that is retained for each segment ***
All accumulated values are up to time of last snapshot of
segment on which the task is observed.
```

- Accumulated problem state CPU time.
- Accumulated supervisor state CPU time.
- Accumulated page I/O's.
- Accumulated pages printed.
- Accumulated MTS I/O counts.
- Accumulated number of responses and total response time.
- Accumulated number of think periods and total think time.
- Virtual memory size summed over observed values from each snapshot in this segment.
- Supervisor estimate of the working set size summed over observed values from each snapshot in this segment.

```
*****
```

FIGURE 6 - Format of a Workload Record

time at the start and the end of the job table scan for each snapshot.

5.2 Evolution of the Workload Model Data Collection

The process of determining what aspects of the workload needed to be captured and how this could be accomplished was extensive. The final design evolved from a number of planned experiments in addition to attempts to correct unsatisfactory results.

5.2.1 Pilot Study I - Determination of Workload Variables

Pilot study I was significantly different from the final experiment in both structure and intent. Thirty two observation periods established at various times of the day and week over a period of one month composed the basis for the study.

The workload monitor used ran for 60 snapshots with a 30 second real time wait between snapshots. The entire 60 snapshots composed one segment. This provided a minimum observation period of one half hour.

At this time none of the stage II variables noted above were available. For the stage I variables the mean of resource consumption across all observed tasks was calculated. Hence, the data was reduced to one set representing an 'average' task. The average number of tasks active at any time was also calculated in order to validate the set of resources considered, a regression model was

constructed with the resource consumptions as the independent variables. The concept was that if all relevant variables had been considered the system performance would be well defined by the resource requests. The set of significant resources could then be reduced via stepwise regression, removing variables which explained little of the variation.

The dependent performance variables taken from the loadlevel records were:

- average percent CPU idle;
- average page I/O's per second to all paging devices;
- average disk I/O's per second;
- average channel activity per second;
- average number of tasks on main CPU queue;
- average class 1 load (a weighted sum of CPU utilization and number of tasks on main CPU queue);
- average class 2 load (a weighted sum of page I/O's to primary and secondary paging devices);
- average class 3 load (number of disk operations);
- average class 4 load (number of I/O operations per channel);
- average class 5 load (number of pages available on primary paging device);
- number of pages a job must have to be considered a big job; and,
- maximum pages allocated to privileged jobs.

The various class loads are the values used to determine external scheduling changes as explained in Chapter 3.

A number of problems became evident. Demands for different resources were highly correlated. There was no one variable available that could be considered a true measure of system performance. Although various load factors were available in the loadlevel data, none were sufficient for the purpose. In particular there was no measure available for response time, which provides a measurement of throughput on an interactive system.

The other concern was that in the regression the variables that were often found to be highly significant were resource variables that in reality should have very little effect due to very low consumption rates. It appeared as if random values explained the deviations more closely than the major resource consumption. It also became obvious that critical resource demands were missing, particularly I/O requests.

From this it was determined that system modifications would be required for any reasonable study and that regression was not an appropriate technique.

5.2.2 System Modifications

The problems encountered indicated that a meaningful study was impossible until measurement modifications could be made to provide the necessary data. Particular examples

make the problem more clear. The only available measure of disk I/O was a system counter that recorded the total number of disk I/O's done by all tasks. Yet one of the more obvious performance problems being encountered was a situation whereby several users would queue for a particular disk drive. If a system model was ever to be able to help solve or explain the problem then detailed data on the I/O per task to the disks would be required.

5.2.2.1 Changes to the MTS Job Program

Two areas needing improvement were:

- I/O measurements; and,
- response and think time measurements.

Changes were made to capture a record of all I/O done by MTS tasks. Space within the MTS system area is allocated to hold this data. Each device type, as defined by MTS, is allocated its own subarea within the larger area, and updating is performed by accessing the area based on the device type index. All devices attached to MTS are classified according to one of the following types:

- unit record;
- magnetic tape;
- terminal;
- disk file;
- paper tape;
- electrostatic printer plotter;
- remote attached device; or

- network.

The only multipurpose type are the unit record devices which include card readers, line printers, card punches, and submission of a batch task from another MTS task.

For each device type five values were retained:

1. number of initializations;
2. number of input operations;
3. total length of data involved in all input operations;
4. number of output operations; and,
5. total length of data involved in all output operations.

The updating of items (2) through (5) is implemented at the MTS-DSR interface. Whenever a line is written or read the appropriate values are updated. Item (1) is the number of times the device has been mounted or opened, depending upon the device in question, and is updated elsewhere.

The decision to collect the requests at the MTS-DSR interface was based upon the discussion of resource units in Chapter 4. Thus the requests could be measured to accurately incorporate the I/O performed without measuring at a system level whereby a simple read or write is converted into a complex string of operations as performed by a lower level system component.

The motivation for recording the response and think times was somewhat different. If a system model is ever to be verified then it is necessary that a definite measure be available for comparing its performance to that of the real system. For an interactive system response time so far is

still the best measure available. Think time measures are necessary if the system, or modelled system, is to be driven in the same fashion as the production system. The collection of these values is implemented at the MTS-DSR interface. Response time is the time interval between the user entering a line and the system responding by either prompting for another line or by generating an output line. Think time is the time interval between the system requesting a line to be entered and reading the line the user has entered, where typing time is included in think time.

5.2.2.2 Changes to Terminal and Network DSRs

The DSR's were extended such that each terminal or network session active during the monitoring period causes the creation of a terminal or network data record. This record retains information on the precise number of I/O's executed by the terminal or network DSR and the accumulated lengths, for both inputs and outputs separately. The necessity for this DSR extension is explained in section 5.1.2.

The DSR modifications to generate these records were implemented by R. Engley to whom I am grateful for his assistance.

5.2.3 Pilot Study II - Determination of Data Collection Interval

Once the required system modifications were

incorporated a decision on an appropriate observation interval was required.

The information retained on each task and its resource consumption is the total amount of the resource consumed during the interval when it is first observed and when it is last observed. Clearly, the longer the interval the more detailed information is being lost on the distribution of the requests by each task. Alternatively, too short an interval provides insufficient time for the system to react or for a representative workload to be observed. In particular, the observation interval needs to be sufficiently long to allow a large number of tasks to go from initiation to termination.

Three one hour intervals were chosen; two during afternoon peaks and one in the early evening. Each hour was spanned by 24 monitor tasks as follows:

- 12 workload monitor tasks lasting for 10 snapshots (5 minutes real time wait) started at 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, and 55 minutes after the hour.
- 6 workload monitor tasks lasting for 20 snapshots (10 minutes of real time wait) starting at 0, 10, 20, 30, 40, and 50 minutes after the hour.
- 3 workload monitor tasks lasting for 40 snapshots (20 minutes of real time wait) starting at 0, 20, and 40 minutes after the hour.
- 2 workload monitor tasks lasting for 60 snapshots (30 minutes of real time wait) starting at 0 and 30 minutes

after the hour.

- 1 workload monitor task lasting for 120 snapshots (60 minutes of real time wait) starting at 0 minutes after the hour.

The number of jobs observed in their entirety for different interval length were:

- 2% to 18% in a 5 minute interval;
- 3% to 16% in a 10 minute interval;
- 18% to 30% in a 20 minute interval;
- 25% to 40% in a 30 minute interval; and,
- 46% to 60% in a 60 minute interval.

A full hour run was required to provide half of the jobs in their entirety. The more tasks that are observed in their entirety, the more closely the data illustrates a typical workload. If the majority of tasks are to be measured in their entirety then a collection session of at least one hour is required.

5.2.4 Pilot Study III - Determination of Intersnapshot Wait Time

A change was made to the snapshot technique. It was felt that any collection session under one hour provided insufficient detail on entire tasks. Therefore a large number of snapshots would be required. However, a longer collection session would only provide consumption values at the beginning of the session or the task and at the end. A 45 minute task begun and completed within the observation

interval would have but one set of values. Tremendous amounts of data would be lost regarding the way in which these resources were consumed. The user may have made all the demands during a 5 minute period and simply left the terminal unattended for the other 40 minutes. Pilot study II had shown that a collection session should be at least one hour in length yet the data loss would be great, providing averages only, with critical peaks smoothed out.

It was necessary to create segments within the collection session so that resource consumption in each segment could be checkpointed. This extension increased the cost of the monitor but provided superior data.

The monitor, extended to handle segments, operated in a fashion similar to the original monitor. Each tenth snapshot, representing 5 minutes of real time wait, as measured from the start-up of the workload monitor, caused an additional data section to be appended to the workload record.

The length of the observation interval was established at 200 snapshots or one hour and forty minutes of real time waits. The actual length of the collection session is dependent on the time it takes the system to perform 200 snapshots. The more users on the system, the more measuring to be done and hence a longer session.

A driver program was written to activate the monitor based on an internal table. Two week cycles were established such that data for nearly every day and hour combination

would be obtained.

The analysis techniques explained in the next chapter were applied to this data. It became clear that a 30 second wait interval between snapshots caused the loss of too much data. In any given segment the workload monitor measured about 10 percent less CPU usage than the loadlevel records reflected. This value peaked to as high as 22 percent during one afternoon collection session. A peak period could take about 120 minutes to complete. In this case each segment was taking about 6 minutes. Each segment represents 10 snapshots so the time was being allocated as 5 minutes of real time waits and 1 minute of processing time. Therefore the time between the subsequent measurement of a given task was about 36 seconds. A series of small batch jobs consuming a relatively large amount of resources could easily come and go within this interval and miss being measured altogether.

The measurement method used in this pilot study had been originally intended as the final study, not a pilot study. However the results invalidated the measurement method. This was unfortunate as it had measured workload during the spring peak (see Figure 4). While the final technique is valid under any load conditions, performance studies are generally concerned with peak conditions. For this reason the loss of valid spring term measurements is regrettable.

The changes made to the monitor included the reduction of the real time wait to 5 seconds. The number of snapshots

was increased to 640, 32 per segment. In this way each segment was constructed of 160 (32×5) seconds of real time wait which added to the expected processing time of 192 seconds (32×6) would produce a run requiring 352 seconds or just 8 seconds short of 6 minutes. The variation on the actual processing interval was now greater as there were a much larger number of snapshots. Given a slack period when processing time was negligible it would now be completed in close to 50 minutes versus the previous case of requiring 100 minutes. The amount of processing to be done had been increased from 200 snapshots to 640 snapshots. The overhead was modified therefore as well by a factor over three. Although in slack periods the monitor could now complete in close to 50 minutes the amount of overhead added during a peak period could extend the collection interval considerably.

6. Conversion and Validation of Workload Data

The initial phase of analysis performs the validation of the resource consumption measured and the conversion of the total resource consumption values to rates of resource consumption.

The data collected is validated by comparing the values measured by the monitor to the values recorded by the SSRTN task in the loadlevel records.

Since the times that each task was first and last observed by the monitor snapshots cannot be expected to coincide with the actual start-up and termination of the task, estimates of the actual arrival and departure times must be analyzed carefully, considering extra resource consumption during unmeasured intervals (see Section 6.3.2 for details).

6.1 Observed Workload Data

Although the collection of data was carefully controlled some unforeseen events did occur. As mentioned in Section 5.2.4 the plan was to run the experiment during the spring term, a peak period of computing activity. As this period represents the heaviest usage period of the year the intent was to collect the desired data during that time and perform the analysis on it. However, as indicated in Chapter 5 this data did not meet the validation standards set and hence was discarded. The monitor procedures were modified as

described in Section 5.1 and subsequently installed in early May, well after the spring term peak.

A further totally uncontrollable event was the upgrade of the AMDAHL V/6 mainframe to the current AMDAHL 470 V/7. Not only was the CPU upgraded, but two additional megabytes of memory were added bringing the total configuration to eight megabytes of memory. The increase in speed of the V/7 CPU over the CPU of the V/6 is usually considered to be about 1.4, but it does vary depending on the particular application. Clearly for ongoing work, and as a present useful application the workload on the current configuration should be emphasized. To assure a rigorous break-in period Computing Services offered CPU time and virtual memory at no charge for the first four days following the upgrade installation. No monitoring was performed during this period as the usage in no way reflected regular production workload.

As implied earlier, throughout the designing stage of the experiment and for the final experiment layout many sets of data observed over the specified collection periods have been analyzed and interpreted at various levels of detail. For succinctness and clarity of this presentation six representative sets of collected data are reported in detail and referred to explicitly throughout the following analysis. The associated times and dates of data collection during the six periods are necessary for understanding the results and interpretations. The periods, in chronological

order, consist of two sessions of relatively high computer usage prior to the installation of the V/7, and four sessions subsequent to the installation of the V/7. The session dates and start-up times are:

- Thursday May 24 at 10 a.m.;
- Thursday June 7 at 2 p.m.;
- Friday June 22 at noon;
- Wednesday June 27 at 1 a.m.;
- Wednesday June 27 at 2 p.m.; and,
- Thursday June 28 at 8 p.m..

The post midnight session and evening session were chosen specifically to ensure that the analysis techniques were equally valid during periods of extremely low usage.

The experiment start-up times, as described in Chapter 5, had originally been designed to provide a comprehensive set of sessions covering various times of the day and week in a regulated fashion. This provided for a variety of sessions, measuring various levels of activity. Originally, it was intended to compare the activities over various sessions across different times of the day and the week. However, such a study is not only curtailed by the volatility of the hardware but also by the periodicity of the natural workload induced primarily by the academic term. This periodicity of system resource demands could be studied by time series methods; however, that would require a much longer timeframe than was feasible for this study.

During the specified measurement periods the hardware

resources varied from day to day due to the failure of specific hardware. Hardware problems similar to this are bound to occur on any system running such a large number and variety of devices. Therefore the chances of sufficient factors remaining constant over a period of time long enough to allow rigorous session to session comparisons are extremely low, and the results of studies which ignore such changes must be treated with caution.

The resource consumption data consists of all non-MTS tasks and all MTS tasks, including the monitor itself. Although the data analysis includes all the tasks, they could be separated out depending on the different specifications of the workload model.

6.2 Execution of the Workload Monitor

With respect to the actual execution of the workload monitor the overhead imposed upon the system by the monitor and the duration of the collection sessions must be discussed in more detail.

6.2.1 Overhead of Workload Monitor

As might be expected the more accurate the data collected by a monitor, the more work the monitor is required to do; and, hence, the effect of the monitor on the system increases. As the intersnapshot interval was reduced from 30 seconds to 5 seconds and the number of snapshots increased from 200 to 640, in order to conserve the same

length of collection session, the overhead of the workload monitor on the system increased by a factor of three. Clearly these two changes in monitoring procedures not only increased the total demands of the monitor but also the rate of its demands. As unfortunate as this may be the data observed prior to these monitor changes was not accurate enough to build a valid workload model and hence the increased monitor demands on the system were a necessity. The main problem with the earlier 30 second intersnapshot interval was that a batch task could easily start and complete within such an interval, thus being totally invisible to the monitor. The error per observed segment, expressed as the percent of total resource consumption measured by the workload monitor, was found to be as high as 22 percent during one peak period and as high as 17 percent during slower periods, which was considered to be much too great to be used as input for building a valid workload model.

The general nature of the collection mechanism is that the higher the load on the system, the higher the demands of the collection mechanism. For instance, the more users on the system, the more virtual memory is required to retain information on the users which in turn increases the paging demands of the monitor program, as its working set size increases. The more users, the greater the difficulty in obtaining this service; yet this is when the monitor requires maximal service. In particular during peak periods

the snapshot exposure time is not negligible with respect to the intersnapshot wait time. In other words the snapshot exposure times increases with load. One experimental run allowed for 640 snapshots representing a total of about 53 minutes of intersnapshot wait time. The actual real time of execution ranged from less than 60 minutes to over 150 minutes (2.5 hours).

It may seem that better, more comparable, results could have been obtained if the real time wait had been associated with a real time interval between consecutive start-up's of snapshot exposures. Upon consideration this was rejected for a number of reasons. If a 5 second start-up to start-up interval had been established, the previous snapshot exposure would often not have been completed by the time the next one was scheduled for commencement. This situation could have been corrected by delaying the next snapshot until the previous one had reached completion. However, irregular observation intervals would again have been the result. Alternatively the intersnapshot interval could have been extended but the information loss due to such a move has already been shown to be excessive.

One of the most important reasons for constant intersnapshot intervals is the concern of minimizing the effect of the monitor upon users. By assuring a 5 second real time wait the monitor goes dormant at regular intervals, making no demands on the system. This is critical especially during afternoon peak periods when system

response is already degraded. Perhaps the cleanest method would be a driver task that initiates snapshot tasks on regular five second intervals regardless of the completion of previous tasks but the degradation would have become intolerable.

As described above the amount of work the monitor must perform is entirely dependent upon the system load. The actual consumption figures are given in Table 1. The numbers illustrate the complexity of the factors that determine the length of the session and the resources consumed by the monitor. For example the June 7th session took longer to complete yet the monitor consumed less CPU time than the May 24th session. The much heavier paging load, as illustrated by the larger number of page I/O's done by all tasks, could easily account for this. For the same amount of productive CPU utilization it could take longer real time, since more paging I/O must be done to bring in the required working set. The monitor consumes less than 6 percent of the CPU time available and during slack periods this value drops to below 1 percent. When the monitor itself accounts for a relatively large portion of the work, as in the June 27th session, it is due to a general lack of work on the system.

In studying these values it must be remembered that the V/7 is faster than the V/6, and that the upgraded system has additional main memory.

Collection Session	Page Reads Done	CPU Time (sec)	Run Time (min)	% Used of Total CPU Time in Run Time
May 24, 10 a.m.				
•by monitor	14818	327.15	105	5.19
•by all tasks	341270	4847.02		76.93
•monitor % of total	4.34	6.74		
June 7, 2 p.m.				
•by monitor	20506	318.63	152	3.49
•by all tasks	613767	6534.20		71.64
•monitor % of total	3.34	4.87		
June 22, noon				
•by monitor	203	109.05	56	3.24
•by all tasks	8907	869.08		25.86
•monitor % of total	2.27	12.54		
June 27, 1 a.m.				
•by monitor	2	25.32	54	.78
•by all tasks	21	97.61		3.01
•monitor % of total	9.52	25.93		
June 27, 2 p.m.				
•by monitor	1186	192.93	60	5.35
•by all tasks	67487	1532.66		42.57
•monitor % of total	1.75	12.58		
June 28, 8 p.m.				
•by monitor	8	39.36	54	1.21
•by all tasks	191	494.87		15.27
•monitor % of total	4.18	7.95		

TABLE 1 - Workload Monitor Overhead

6.2.2 Analysis of Snapshot Exposure Times

The actual elapsed run time session length is directly related to the actual duration or exposure time of each snapshot. Except for the two V/6 data sets the session length (see Table 1.) exceeded the total real time waits by 7 minutes or less, thus leaving less than about .75 seconds for each snapshot, a relatively insignificant amount compared to the real wait time. Therefore exposure plus wait times give an interval of less than 6 seconds between consecutive observations of each task. This, however, is not the case with the V/6 data. The June 7th session had snapshot exposure times ranging from 0 to 46 seconds with a mean of over 9 seconds. In this case the snapshot exposure time is no longer insignificant. Assuming that a given task is measured at the same point in each snapshot then there are now 14 seconds between updates. This assumption is reasonable since the job table is always scanned in the same direction starting at the same location. However, it is not a precise analysis as the duration of each snapshot varies considerably about the average. A task could, therefore, be measured on a particular snapshot, continue to consume resources for the next 13 seconds and terminate prior to being measured on the subsequent snapshot leaving all consumption during the final 13 seconds unmeasured. To minimize this potential error the intersnapshot interval needed to be reduced as much as possible and hence the 5 second value. Further error adjustments are described in

Section 6.3.1.

Another point is worthy of discussion. No changes were made to the collection technique with the installation of the V/7. Recall that the number of snapshots to be taken was based upon a desired collection interval of 120 minutes during low peak periods. This was established from the results of pilot study III. During these periods it was determined that each snapshot would take about 6 seconds of real time. This value, as mentioned earlier, is load dependent and also necessarily machine dependent. A period of peak usage on the V/7 would be required to decide if the new machine significantly reduces this value, making monitor modifications necessary. The data available is inconclusive as it does not include peak periods during the regular academic term. Although the V/7 is more powerful it is more than likely that the ever increasing user load will be sufficient by fall to retain the exposure time at about that measured on the V/6 during the previous spring term. This poses no real problems but is merely a consideration to be kept in mind with regard to changing workload level and machine capability.

6.3 Data Conversion

The data conversion stage has a primary purpose of converting the total resource consumption data to resource consumption rate data. Since a rate is merely the total consumption per unit time, these rates can only be

calculated once the lifetime of the task has been established and the error alluded to above has been incorporated. Once the lifetimes and actual resource consumption rates have been determined for each task, the total consumption figures can be attained for the subsequent data validation. The actual lifetime of a task in a segment starts when either the task is initialized by the supervisor, referred to as the arrival of the task, or the start of the first snapshot, whichever occurs last. The actual lifetime ends at either task termination, referred to as the departure of the task, or the end of the wait interval prior to the snapshot that marks the end of the segment, whichever occurs first. This is the actual lifetime as opposed to the observed lifetime. The observed lifetime is the interval from the time the task is first measured by a snapshot to the last time it is measured by a snapshot. The adjustment for the difference in the observed and actual lifetimes is discussed in Section 6.3.1.

The actual resource consumption rates as generated by the data conversion analysis are listed in Figure 7. Since all values are retained internally as integers some of the values are multiples of 1000 to retain the significant digits. The resource consumption rates are denoted as

$CR(i,j)$ - the consumption rate of task i of resource j calculated as the adjusted actual consumption divided by the adjusted actual lifetime of the task.

This data is generated for each task during every segment in


```

*****
***** Identification *****
• UMMPS task number.
• Task type (i.e. PDP, HASP, MTS, etc.).
• If MTS task then CSid and type: terminal, batch, or
  system.

*****
*****Data Section that is retained for each segment*****
• Starting snapshot (0<snapshot<32).
• Ending snapshot (32 implies it continued into next
  segment).
• Average number of pages of virtual memory per snapshot.
• Average of the supervisor's estimate of the working set
  size per snapshot.
• Average problem state CPU consumed per second in 1000ths
  of a microsecond.
• Average supervisor state CPU consumed per second in
  1000ths of a microsecond.
• Average number of page I/O's done per second times 1000.
• Average number of HASP pages printed per second times
  1000.
  For each of unit record devices, magnetic tape drives,
  terminals, disk files, paper tape, electrostatic printer
  plotter, remote attached devices, and network devices
  the following 5 fields appear:
=====
• Average number of initializations per second times 1000.
• Average number of input operations per second times
  1000.
• Average total data length involved with input operations
  per second times 1000.
• Average number of output operations per second time
  1000.
• Average total data length involved with output
  operations per second times 1000.
=====
• The total number of think times measured in the segment.
• The total time due to think times in the segment
  measured in 1/122nds of a second.
• The total number of response times measured in the
  segment.
• The total time due to response times measured in the
  segment measured in 1/122nds of a second.

```

FIGURE 7 - Format of Resource Consumption Rate Vector

which the task was active. The first segment extends from the beginning of the first snapshot, snapshot 0, through to the beginning of snapshot 32. This period includes 32 snapshots and 32 intersnapshot real time wait periods, see Figure 5. The second segment extends from the start of snapshot 32 through to the start of snapshot 64. As the analysis progresses sequentially a segment at a time, the snapshots within each segment will be referred to as snapshots 0 through 31 for any segment.

6.3.1 Measurement Error

The error due to taking discrete measurements via snapshots was introduced above. It can occur as follows:

1. When a task arrives between snapshot i and snapshot $i+1$, its consumption from arrival to snapshot $i+1$ cannot be measured.
2. When a task departs between snapshot j and snapshot $j+1$, its consumption from snapshot j until departure cannot be measured.
3. When a task arrives and departs between snapshot i and snapshot $i+1$, it is not visible to the monitor.

Two different types of tasks exist and must be handled differently. Recall that a task, as far as the monitor is concerned, is a match on the task number and, in the case of MTS tasks, an additional match on the CSid. In the case of non-MTS tasks and batch MTS tasks a clean job table is obtained for each new task or user session as viewed by the

monitor. In this case each new task to the monitor is also a new task to the supervisor. This will be referred to as a type Y task.

On terminal devices however, it is possible that a number of different users may use the same terminal in such a way that the MTS task number remains the same, the same job table remains in use and the values for CPU and page I/O usage simply get accumulated throughout the day. For example if a terminal is used for a session by a particular user and then another user uses the same terminal, the monitor records two tasks, while the supervisor treats it merely as a continuation of the original task and hence the job table values are not initialized. This type of MTS task will be called a type Z task.

For type Z tasks no values are available for total consumption from arrival until the first snapshot measurement is made by the monitor. With type Y tasks the accumulated values stored within the internal system structure can be entirely attributed to this task. Therefore, if it is first observed on snapshot i consumption to this point is known precisely and error (1) is eliminated. For type Z tasks however the consumption to the first observation snapshot must be estimated, as shown in the analysis below. Because all terminals have the ability to support type Z tasks, and it is impossible to determine whether or not they actually were Z tasks, all MTS terminal tasks are assumed to be type Z, and all other tasks are

taken to be type Y.

Another variant of the collection mechanism was tried whereby rather than matching on UMMPS job number and CSid, only the jcb number was considered in an effort to remove all type Z tasks. Additionally, rather than considering only active tasks, tasks inactive but attached to terminal lines were included. Under this mechanism a task attached to a terminal would be recorded as one task by the monitor as long as the supervisor regards it as one task also. This scheme made all tasks type Y and hence eliminated error (1). In theory it was a cleaner system, in practice it failed. At any one time there can be in the order of 150 of these inactive terminal tasks, greatly increasing the load on the monitor and its virtual memory requirements. Although these tasks are doing no work the monitor updates and processes them in the same fashion as the active tasks. The overhead increased greatly, for example one session run in this fashion, during a period when the other monitor would have completed in two hours, required over four hours to run!

6.3.2 Error Adjustments

In general terms the collection mechanism took $S+1$ snapshots, numbered 0 through S , of the system during each segment where S is 31 and $S+1$, 32, is snapshot 0 of the next segment. All times are given in seconds measured from an initial date and time reference.

Detailed term definitions are required in describing

the correction of the above errors. The first 10 terms are with respect to times and intervals, the last 4 with respect to resources and all refer to any one segment.

- $T(j)$ for $0 \leq j \leq S+1$

The snapshot exposure time of snapshot j .

- $N(j)$ $0 \leq j \leq S+1$

The number of unique tasks observed during snapshot j .

- M

The total number of unique tasks observed during all snapshots, 0 through $S+1$.

- $t(i,j)$ for $0 \leq j \leq S+1$, $1 \leq i \leq N(j)$

The time at which task i was processed during snapshot j . This value is only available for the first snapshot during which the task was observed.

- $TMIN(j)$ for $0 \leq j \leq S+1$

The minimum time over all $t(i,j)$ for $1 \leq i \leq N(j)$.

The time at which the job table scan started for snapshot j .

- $TMAX(j)$ for $0 \leq j \leq S+1$

The maximum time over all $t(i,j)$ for $1 \leq i \leq N(j)$.

The time at which the job table scan was completed for snapshot j .

- $TMID(j)$ for $1 \leq j \leq S+1$

The midpoint of $TMIN(j)$ and $TMAX(j)$.

- $IMID(j,j+1)$ for $1 \leq j \leq S$

The midpoint of $TMID(j)$ and $TMID(j+1)$.

- $A(i)$ $1 \leq i \leq M$

The time at which task i arrived, the beginning of its actual lifetime. Note that this time is not available.

- $D(i) \quad 1 \leq i \leq M$

The time at which task i departed, the end of its actual lifetime. Note that $D(i) \geq A(i)$. This time is also not available.

- X

The number of unique hardware resources.

- $R(i, j, x) \text{ for } 1 \leq i \leq M, 0 \leq j \leq S+1, 1 \leq x \leq X$

The amount of resource x consumed by task i up to $t(i, j)$.

- $C(i, x) \text{ for } 1 \leq i \leq M, 1 \leq x \leq X$

The amount of resource x consumed by task i between $TMIN(0)$ and $TMIN(S+1)$, its actual lifetime consumption in the segment.

- CF

The ratio of the actual lifetime of the task to its observed lifetime. The actual lifetime is the time from $A(i)$ to $D(i)$. The observed lifetime is from $t(i, h)$ where task i is observed during $T(h)$ but not during $T(h-1)$ until $t(i, g)$ where it is observed during $T(g)$ but not during $T(g+1)$.

The real lifetime of each task in each segment can be categorized as one of the following five types:.

1. $A(i) \leq t(i, 0), D(i) \geq t(i, S+1)$
2. $A(i) \leq t(i, 0), D(i) < t(i, S+1)$
3. $t(i, 0) < A(i) \leq t(i, S+1), D(i) \geq t(i, S+1)$

4. $A(i) > t(i,0)$, $D(i) < t(i,S+1)$
 5. $A(i) > t(i,j)$, $D(i) < t(i,j+1)$ for some j such that $1 \leq j \leq S$.
- These will be called the lifetime categorizations. These categorizations are illustrated in Figure 8.

All resource consumptions between $TMIN(0)$ and $TMIN(S+1)$ are to be measured. Given task i this means measuring the resource consumption from the later of $TMIN(0)$ or $A(i)$ to the earlier of $TMIN(S+1)$ or $D(i)$.

Because $A(i)$ and $D(i)$ do not coincide with any of the $t(i,j)$ for $0 < j \leq 32$ it is necessary to extrapolate the measured resource consumption to the interval not observed. Five basic assumptions are used to aid in estimating these measurement holes.

1. If a task was last observed during $T(j)$ and then observed missing during $T(j+1)$ it in fact departed at $IMID(j,j+1)$. Similarly if a task is first observed during $T(j+1)$, then it arrived at $IMID(j,j+1)$. In type Y tasks consumption is known from $A(i)$ to $T(j+1)$ to be the accumulated values observed at $T(j+1)$. For type Z tasks only the data observed from $T(j+1)$ on is usable.
2. If a task i existed at time $t(i,0)$ it existed at $TMIN(0)$.
3. If a task existed outside of the time interval for which measurements are available, it consumed resources at the same consumption rate outside as during the observed lifetime in the segment.
4. Because $t(i,j)$ is only retained for the first snapshot

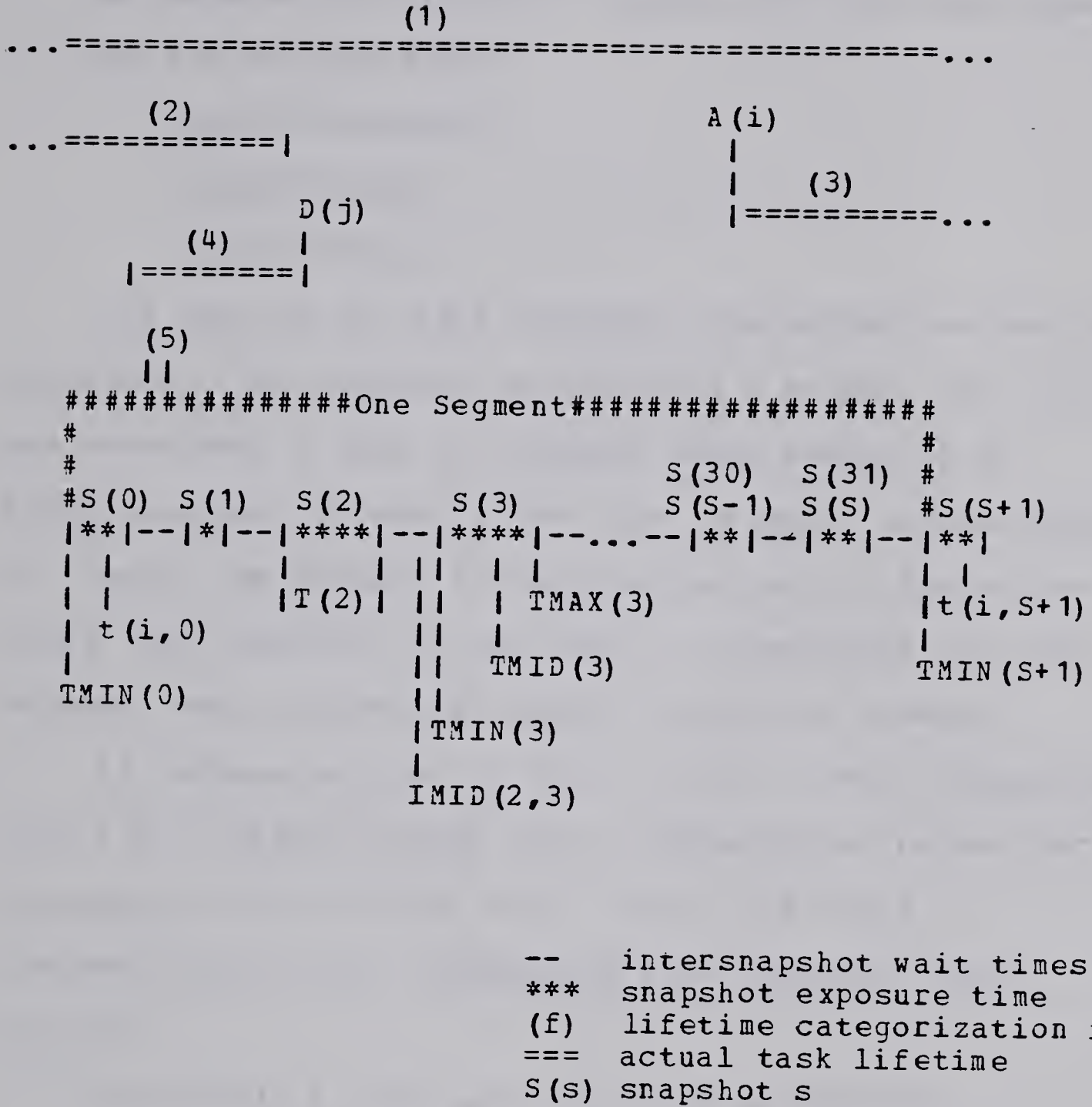


FIGURE 8 - Structure of a Segment

during which task i is observed, an estimated $t(i,j)$ may be required. If $t(i,S+1)$, $t(i,0)$, or $t(i,j)$ are required and not available use

$$t(i,S+1) = TMID(S+1)$$

$$t(i,0) = TMID(0)$$

$$t(i,j) = TMID(j)$$

Now each of the five lifetime categorizations can be considered. The analysis is performed a segment at a time, and therefore, a task for example could easily be of categorization (3) during the first segment, categorization (1) during the second, and categorization (2) during the third. The behaviour of the task is categorized for each segment, and not over the entire collection session.

In categorization (1) $C(i,x)$ is the total consumption from $t(i,0)$ until $TMIN(S)$ plus a compensation factor for the consumption from $TMIN(0)$ until $t(i,0)$ and minus a compensation for the consumption from $TMIN(S+1)$ until $t(i,S+1)$.

$$C(i,x) = [R(i,t(i,S+1),x) - R(i,t(i,0),x)] * [1 + CF]$$

where

$$CF = \frac{[t(i,0) - TMIN(0)] - [t(i,S+1) - TMIN(S+1)]}{t(i,S+1) - t(i,0)}$$

In categorization (2) the task is last observed at $t(i,j)$ yet has departed before $t(i,j+1)$. Therefore the resource consumption used is that from $t(i,0)$ until $t(i,j)$ plus compensation factors for $TMIN(0)$ until $t(i,0)$ and

$t(i, j)$ until $D(i)$.

$$C(i, k) = [R(i, t(i, j), x) - R(i, t(i, 0), x)] * [1 + CF]$$

where

$$CF = \frac{[t(i, 0) - TMIN(0)] + [IMID(j, j+1) - t(i, j)]}{t(i, j) - t(i, 0)}$$

It is possible that such a categorization (2) task departed between snapshot 0 and snapshot 1, then for the interval $TMIN(0)$ to $IMID(0, 1)$ the resource consumption rates are assumed equal to the respective values in the last segment.

In categorization (3) the task is first observed at $t(i, j)$, where $t(i, j-1) < A(i) \leq t(i, j)$. In this case type Y and type Z tasks must be handled differently. For type Z tasks the extra adjustments required include the consumption from $A(i)$ to $t(i, j)$ and to exclude the consumption from $TMIN(S+1)$ to $t(i, S+1)$.

$$C(i, x) = [R(i, t(i, S+1), x) - R(i, t(i, j), x)] * [1 + CF]$$

where

$$CF = \frac{[t(i, j) - IMID(j-1, j)] - [t(i, S+1) - TMIN(S+1)]}{t(i, S+1) - t(i, j)}$$

For type Y tasks the extra adjustment required is to exclude the consumption from $TMIN(S+1)$ to $t(i, S+1)$. $A(i)$ is estimated to be $IMID(j-1, j)$ but the related resource consumption is known.

$$C(i, x) = [R(i, t(i, S+1), x) - R(i, A(i), x)] * [1 + CF]$$

but $R(i, A(i), x)$ is zero by definition therefore

$$C(i, x) = R(i, t(i, S+1), x) * [1 + CF]$$

where

$$CF = \frac{-[t(i, S+1) - TMIN(S+1)]}{t(i, S+1) - IMID(j-1, j)}$$

It is possible that the task is first observed on snapshot $S+1$. For type Y tasks this poses no difficulties. For type Z tasks it means no data is available on the consumption during the segment in which $A(i)$ occurred. The solution is that processing continues in the next segment and the resources consumed in it can be applied to the previous segment as mentioned in assumption 3. above.

In categorization (4) for type Z tasks both $A(i)$ and $D(i)$ must be approximated by midinterval values, and for type Y tasks only $D(i)$ needs to be approximated. Suppose that for task i

$$t(i, j) < A(i) \leq t(i, j+1)$$

and

$$t(i, m) \leq D(i) < t(i, m+1) \text{ where } 0 \leq j < m \leq S+1$$

then for type Z tasks

$$C(i, x) = [R(i, t(i, m), x) - R(i, t(i, j+1), x)] * [1 + CF]$$

where

$$CF = \frac{[t(i, j+1) - IMID(j, j+1)] + [IMID(m, m+1) - t(i, m)]}{t(i, m) - t(i, j+1)}$$

and for type Y tasks $A(i)$ is estimated to be $IMID(j, j+1)$ and

$$C(i, x) = [R(i, t(i, m), x) - R(i, A(i), x)] * [1 + CF]$$

but $R(i, A(i), x)$ is zero by definition, therefore

$$C(i, x) = R(i, t(i, m), x) * [1 + CF]$$

where

$$CF = \frac{IMID(m, m+1) - t(i, m)}{t(i, m) - IMID(j, j+1)}$$

The invisibility of categorization (5) tasks poses difficulties as mentioned earlier.

There is a special case of categories (2), (3), and (4) that is treated quite differently for type Y and type Z tasks. Consider task i first observed on snapshot j and then observed missing on snapshot $j+1$. For type Y tasks the consumption from $A(i)$ to $t(i, j)$ is known and can be used to estimate consumption from $t(i, j)$ to $D(i)$. However for a type Z task no information is available and its consumption is lost. Batch tasks which are most likely to fall within this category are type Y tasks by definition and hence are accounted for. The fact that the measured data can be validated shows that very few tasks are escaping measurement altogether.

Thus this detailed procedure of accounting for all possible resource consumption error ensures the validity of the data (see Section 6.4.2).

6.3.3 Software Tools

Before the actual conversion to consumption rates can be done the terminal and network DSR adjustments must be made. The two programs involved in this are discussed below.

6.3.3.1 DSR Record Reduction

The preliminary step involves correcting the terminal and network I/O as measured by the monitor.

A DSR record is created in the DSRSTAT file for all terminal or network connections during a period for which the system switch is set (recall Section 5.2.2.2). These are the resources for which it is possible to generate usage which is not visible to MTS.

These records are written out when the terminal task terminates, the network call completes, or when terminal or network I/O is attempted and the switch has been reset. Clearly DSR records do not necessarily appear in the same order within the DSRSTAT file as the associated monitor records in the QQ file.

An assembler program was written to extract the required information from the DSRSTAT file and produce an output file with new compensation records ordered according to the order of appearance in the QQ file.

The input to the program consists of:

1. QQ file
2. DSRSTAT file

The output from the program consists of:

1. file to which DSR compensation records are written

The data retained in the output file from this program is:

- task number;
- time at which DSR started collecting statistics for this task;
- time at which DSR stopped collecting statistics for this task;
- type (3270, FECP, Network); and,
- compensation factors for
 - total inputs
 - total input length
 - total outputs
 - total output lengths

where the compensation factors are calculated as

$$(\text{total measured by DSR} / \text{total measured in QQ file}) * 1000.$$

The program operates by scanning through the QQ file for each consumption vector that recorded terminal or network activity. The DSRSTAT file is searched for a matching DSR record, and then the compensation factors are calculated and written in the compensation record.

6.3.3.2 Workload Conversion

The workload conversion program is also written in assembler. As input it takes the QQ file and the output from

the DSR reduction program and generates information on the total resources consumed during each segment and the resource consumptions rates for each segment.

The input to the program consists of:

1. the QQ file; and,
2. the output file 1. from the DSR record reduction.

The output from the program consists of:

1. a rate of resource consumption vector for each task in each segment;
2. $TMIN(0)$, $TMIN(S+1)$, and total adjusted CPU and page I/O usage for each segment as measured by the workload monitor after applying the error correction techniques; and,
3. start and end times for each segment as well as the breakdown on the duration of the tasks over the entire collection interval divided into MTS and non-MTS tasks.

The calculations performed are those described in detail in Section 6.3.

6.4 Data Validation

It is essential that all resources consumed be recorded, hence, a check on the above procedure is performed for CPU usage and paging activity. The total system wide consumption from the loadlevel records is compared to the total resource consumption calculated by the workload conversion routine. Each segment individually, and the entire collection interval are all verified.

Minor adjustments need be made to the loadlevel records to obtain the overall system demands between $TMIN(0)$ and $TMIN(S+1)$, as chances are that the loadlevel records were not generated at precisely $TMIN(0)$ and $TMIN(S+1)$. Recall that they are generated once every 20 seconds. The algorithm to adjust for this proceeds as follows. Let, for example, time y be the time stamp on the first loadlevel record generated with a time stamp value of greater than $TMIN(0)$ and let time z be the time stamp value on the last loadlevel record generated before $TMIN(S+1)$. Resource consumption from the loadlevel records is initially calculated as the accumulated usage to z minus the accumulated usage to y . Because y and $TMIN(0)$, and z and $TMIN(S+1)$ do not coincide the loadlevel value is multiplied by

$$1 + \frac{[y - TMIN(0)] + [TMIN(S+1) - z]}{z - y}$$

to adjust the actual consumption in $(TMIN(0), TMIN(S+1))$. Because $y - TMIN(0)$ and $TMIN(S+1) - z$ are both necessarily less than 20 seconds, the total length of time involved in the estimation is small, and hence so is any error introduced.

6.4.1 Verification Software

The verifier program, for the above procedure is written in assembler. The system loadlevel records for a period which includes the interval from $TMIN(0)$ to $TMAX(640)$

are collected as discussed in Section 5.1.1.

The input to the program consists of:

1. the loadlevel records; and,
2. the output file 2. from the workload conversion run.

The output from the program consists of:

1. Detailed data for each segment in addition to that of the entire collection interval. The data includes the actual start and end times of the segments versus the times of the closest loadlevel records, the values for CPU and page I/O usage from the two sources and the percentage error. The number of jobs in each lifetime categorization are also given.

6.4.2 Validation Results

The error is consistently less than 5 percent for all segments for both CPU time and page I/O's. Although the error generally indicates that the monitor has missed consumption which the error analysis techniques have not entirely accounted for, occasionally the direction of the error is reversed (not in any of the given cases). The results of the chosen collection sessions are given in Table 2. Where the error exceeds 5 percent the system is relatively inactive; and, in fact the larger percentage error represents less absolute error.

The first section of Table 3 indicates the number of tasks that were in each lifetime categorization. The numbers in each category are the values in each category summed over all segments. For example a single task that was category

Collection Session	May 24 10 a.m.	June 7 2 p.m.	June 22 noon	June 27 1 a.m.	June 27 2 p.m.	June 28 8 p.m.
--------------------	-------------------	------------------	-----------------	-------------------	-------------------	-------------------

Actual Page I/O's	341270	613767	8907	21	67487	191
-------------------	--------	--------	------	----	-------	-----

Measured Page I/O's	334976	605687	8157	18	65391	47
---------------------	--------	--------	------	----	-------	----

% Error Page I/O's	1.84	1.31	8.42	14.28	3.10	75.39
--------------------	------	------	------	-------	------	-------

Actual CPU (sec)	4847.02	6534.20	869.08	97.61	1532.66	494.87
------------------	---------	---------	--------	-------	---------	--------

Measured CPU (sec)	4652.26	6219.40	824.41	90.77	1480.21	488.93
--------------------	---------	---------	--------	-------	---------	--------

% Error CPU	4.01	4.81	3.06	7.00	3.42	1.20
-------------	------	------	------	------	------	------

TABLE 2 - Collection Error Results

Collection Session	May 24 10 a.m.	June 7 2 p.m.	June 22 noon	June 27 1 a.m.	June 27 2 p.m.	June 28 8 p.m.
# Tasks ¹ in Lifetime Category (1)	2700	2685	2196	893	3172	1172
# Tasks ¹ in Lifetime Category (2)	391	542	156	22	262	37
# Tasks ¹ in Lifetime Category (3)	421	477	119	19	228	39
# Tasks ¹ in Lifetime Category (4)	154	276	52	20	89	11
# Tasks Observed in Collection Session	752	965	322	88	515	114
# Tasks Observed in Entirety	494	712	134	37	266	34
% Observed in Entirety	65.7	73.7	41.6	42.0	51.6	29.8

¹ - summed over all segments

TABLE 3 - Lifetime Categorization Results

(3) in segment 1, category (1) in segments 2 through 19, and category (4) in segment 20, would account for 1 of each of the category (2) and category (3) count, and 18 of the category (1) count. The second section of Table 3 gives the number of unique tasks observed over the entire collection session, where a single task observed across several segments counts as one task, and the number of these that arrived after the first snapshot of the initial segment and departed prior to the final snapshot of the last segment are those tasks observed in their entirety. As described in Section 5.2.3 a high percentage of tasks observed from arrival to departure assures a representative measure of true system workload. During heavy periods, when the monitor ran for at least one hour, at least one half of the tasks observed are of this type. During low periods however, very few users are arriving and departing, compared to the number of system tasks running, so the percentage falls. System tasks are those tasks that run as a general operations procedure and consist of all non-MTS tasks in addition to the system MTS tasks as described in Section 3.3. The low percentage on the June 22nd noon hour run, reflects that few users actually begin and end session over the lunch break. Therefore during heavier periods, when the monitor takes over one hour to complete, at least 50 percent of the tasks are observed in their entirety.

With the construction of segments within the collection session, the collection session is therefore long enough to

observe most tasks through from arrival to departure, while in any given segment the majority of tasks are category (1). This reduces the measurement error since, generally speaking, the period of time to which the error adjustments must be applied is less for category (1) tasks than any other category. For example the time from the middle of a snapshot to the start or end of it, versus the time from the middle of the intersnapshot interval to the middle of the snapshot.

6.5 Converted and Validated Data

The analysis to this point provides for each segment:

1. the consumption rate, $CR(i,j)$, of each task i of every resource j , $1 \leq j \leq X$; and,
2. the time period (the arrival and departure times) over which each task consumed the resources at these rates.

The consumption rate for each task is the observed consumption divided by the observed lifetime, which is equal to the adjusted consumption divided by the adjusted lifetime.

As section 6.2.3 shows the analysis is performed such that there are only $S+3$ time points when a task can arrive or depart within a given segment. These contiguous time intervals, called 'model intervals', built on these $S+3$ time points are the ones that would be supplied to the system model which would schedule various activities during those intervals. The $S+3$ time points at which modelled arrivals

and departures can occur are:

(1) $TMIN(0)$

(2) $IMID(0,1)$

(3) $IMID(1,2)$

•

•

•

(S+2) $IMID(S,S+1)$

(S+3) $TMIN(S+1)$.

Therefore each segment is composed of the following S+2 model intervals:

(1) $[TMIN(0), IMID(0,1)]$

(2) $[IMID(1,2), IMID(2,3)]$

•

•

•

(S+1) $[IMID(S-1,S), IMID(S,S+1)]$

(S+2) $[IMID(S,S+1), TMIN(S+1)]$

The lifetime of any task observed in a segment can be denoted by the series of model intervals during which it was active.

The subsequent step in the analysis is to determine classes of tasks within each segment such that the workload model need only supply parameters on a small set of classes of tasks, and details on the number of active tasks in each class during any model interval.

7. Construction of Workload Model

The analysis described in Chapter 6 provides for each segment the resource consumption rates for each task and the model intervals during which it used the corresponding resources. The analysis in this chapter characterizes the active tasks in each segment as classes of tasks, or task groups which form the basis of the actual workload model. Each such group is then typified by the average resource consumption rates of its constituent members. During any model interval the number of members in each task group can be established from the number of active tasks in that interval. These task group sizes are further parameters used in the workload model.

7.1 Determination of Task Groups

The specific resource consumption data that is collected, while precise, tends to be too voluminous. The requirement to condense this data, in some meaningful way, results in attempting to group the tasks according to specific characteristics. Hence, specific task groups are crystallized out of the total data and then only the specific parameters of the task groups need to be considered in the workload model.

Previous workload characterization studies have tended to produce classifications via manual techniques. The classification of user types for VM/370 by Bard (BARD76a,

BARD76b) is done manually and relied upon the particular scheduling mechanism employed by VM/370. In his paper Bard suggests that should the number of users to be classified approach 100, automatic techniques would need to be used. He represents each user class by the mix of trivial and nontrivial transactions with the average resource consumption rates calculated over all transactions of that type over all constituent users.

While the intent here is to produce a similar type of classification based on resource consumption rates the number of tasks involved well exceeds the suggested value of 100. Thus a systematic grouping procedure is necessary and cluster analysis is a useful tool. The use of cluster analysis to determine user groups was first presented by Hunt (HUNT71), although actual rigorous application of cluster analysis in this area did not appear until the initial work by Agrawala et al. (AGRA76) and that of Fangmeyer (FANG76). The work of Fangmeyer adds little to the original approach taken by Hunt and makes no attempt to characterize the workload as a whole, but rather uses it as a tool to study the respective usage of, say, two resources.

Agrawala, however, develops a much more detailed technique designed specifically for the workload characterization application. The technique employed here is a slightly modified adaption of Agrawala's approach. The intent of Agrawala's work is somewhat different but, with some modifications, the techniques are applicable in the

present study. The goal, in Agrawala's study, is an overall classification of the types of users running on the UNIVAC 1108 EXEC 8 system at the University of Maryland over a one to three day period. As Agrawala mentions, the concern underlying the use of the clustering technique is that, although the degree of variation in the whole user population is large, well defined user subclasses exist within it. A very complicated probability distribution representing the demands of the total user population can thus, be replaced by several simpler distributions. Because it is difficult, if not impossible, to ascertain the actual conditional distribution describing the total user population, the technique used is based on no a priori knowledge of that distribution. The Agrawala clustering study is based on the total resource demands of jobs measured over a selected period, the intended purpose being to reconstruct a representative test workload. Clusters are found to exist in their study but, similar to all educational institutions, are found to be highly variable with the time of year.

The current application is different in that a more precise model is determined. The computer system involved in the current study is considerably more complex, and although Agrawala attempts clustering by different methods, including total resource consumption and number of runs, resource consumption rates are never considered. The difference in the application is best brought out by considering a change

that is required with regard to the clustering algorithm (AGRA76). In their case if a user group or cluster has less than 5 users, the data for these users is removed from the original data set and the algorithm continues with the reduced data set. However, these 5 users could represent 99 percent of the work on the system, and yet because they form a small group are discarded. Such a technique is entirely contrary to the current purpose.

The analysis by Bard results in 3 basic task or user groups where if any single user group represents too wide a variety of usage then it can be increased to 2 or more groups. Although 3 groups are probably too few natural groupings to be expected to characterize a workload it indicates the type of classification desired. The requirement is to condense the data as much as possible while retaining as much information as possible. For example, it is of minimal use if the classification scheme reduces 100 users to 50 task groups. A broader classification of at most 10 task groups would be more useful for the intended application.

The use of clustering techniques allows natural task groupings to evolve. Input parameters to the clustering algorithm require experimentation to determine the most useful constraints and coefficients for the current application. For example these values could be set in such a way as to force all tasks into the same task group or to allow each task to form its own task group, quite contrary

to the objective.

7.1.1 Clustering Algorithm

The clustering technique is a simple process based on the k-means algorithm (ANDE73), where each of k resources is represented by an axis in a k-dimensional space. A number of cluster centroids are initially selected. Each data point is subsequently assigned to the cluster that minimizes the distance measure between it and the centroid. Subsequent passes, or iterations, through the data points are used to determine if clusters should be split or joined and to which cluster each point should be moved.

The distance function is a weighted Euclidean distance. The weighting factors are initially set to a constant but on subsequent iterations will change depending on the variance of a given variable within a specific cluster. A variable that well describes the membership of a cluster will have a small within cluster variance. The weight assigned to the axis for this variable for that specific cluster is related to the inverse of the variance of this variable. Therefore, once an axis of a variable with small variance has emerged as a good determining cluster factor other potential cluster members will necessarily have to be close to the cluster centroid along that axis. The opposite applies to those variables that have a large within cluster variance.

In order to describe this particular clustering technique and illustrate its applicability in classifying

user or task groups several definitions and notations, in addition to those given in Chapter 6, must be specified:

- X' is the number of resources to be used in determining the task groups where $X' \leq X$. For ease of use, the original X resources are reordered so that the first X' resources are those to be used in clustering.
- $CM(k, x)$ is the mean of the k 'th cluster with respect to resource x , $1 \leq x \leq X'$.
- $CM(k, *)$ is the mean vector of the k 'th cluster, which is the cluster centroid.
- $W(k, x)$ is the weighting factor to be associated with resource x in the k 'th cluster, $1 \leq x \leq X'$.
- $W(k, *)$ is the weighting vector of the k 'th cluster.
- WC is a small constant to be used in calculating the $W(k, x)$.
- $VR(k, x)$ is the estimated value of the variance of resource x within cluster k , $1 \leq x \leq X'$.
- $CR(i, j)$ is the consumption rate (mentioned in Chapter 6) of task i of resource j calculated as the adjusted actual consumption divided by the adjusted actual lifetime of the task. These are the values already calculated for all j such that $1 \leq j \leq X$. All subsequent analysis is only applied to the resources included in X' and x is used rather than j as the variable index.
- $CRL(i, x)$ is the natural log of $[CR(i, x) + 10]$, $1 \leq x \leq X'$, the additive constant being required because $CR(i, x)$ can be 0.

- $CRS(i, x)$ is $CRL(i, x)$ scaled.
- $CRS(i, *)$ is the vector of scaled resource consumption rates for task i . This constitutes a data point in X' dimensional space.
- $D(CM(k, *), CRS(i, *))$ is the dissimilarity measure of task i from cluster k where task i has consumption rate vector $CRS(i, *)$ and cluster k is located at the centroid $CM(k, *)$.
- NC is the current number of clusters.
- $K(k)$ is the number of data points in cluster k (cluster members).
- $S(k, x)$ is the current sum of the scaled resource consumption rates of resource x over all members of cluster k , $1 \leq x \leq X'$.

For meaningful results all X' resources must be measured in similar units and on the same scale. A simple way to do this is to divide each observed value for resource x by the largest value observed for resource x , placing all values in the range of 0 to 1. The problem with this method is that a single large value, widely separated from the rest of the other values, will greatly distort the other values after scaling. The scaling technique used here is similar to the above but does allow for outliers, by excluding them in scaling the rest of the values.

The values to be clustered, the $CR(i, x)$ for $1 \leq x \leq X'$, are converted to logs, $\ln[CR(i, x) + 10]$, in order to reduce the spread of these values. The scaling proceeds further by

determining H such that 97.5% of the $CRL(i, x)$, taken over all tasks $1 \leq i \leq M$ for a given resource x , are less than $CRL(H, x)$ and L such that 97.5% of the $CRL(i, x)$, taken over $1 \leq i \leq M$, are greater than $CRL(L, x)$, where the interval $[CRL(L, x), CRL(H, x)]$ contains 95 percent of the log resource consumption rates for a given resource x . A new scaled $CRL(i, x)$, called $CRS(i, x)$, is calculated as

$$CRS(i, x) = \frac{10 * (CRL(i, x) - CRL(L, x))}{CRL(H, x) - CRL(L, x)}$$

which for any given resource x are values between 0 and 10. These $CRS(i, x)$'s from now on are referred to as the values of variable x , the (scaled) resource (x) consumption rates. Hence, most of these values lie within a hypercube situated at the origin with sides of length ten.

Initial cluster centroids for m clusters are chosen by any of a number of techniques. The more common techniques are to choose m data points from the actual data or, alternatively, to establish m points that span the entire range. The actual techniques used are described in Section 7.1.2.

The dissimilarity measure between the data point for task i and the centroid of cluster k , based on simple Euclidean distance, is

$$D[CM(k,*), CRS(i,*)] = \sqrt{\sum_{x=1}^{X'} (CM(k,x) - CRS(i,x))^2}$$

The extension to this measure is to weight each term $[CM(k,x) - CRS(i,x)]^2$, corresponding to task i , by the inverse of the within cluster k variance along the axis associated with the resource x in question. This inverse variance value is normalized by the sum of the weights. Due to this normalization the dissimilarity measures between a given data point and any two clusters with very different within cluster variances can then be compared.

Hence

$$D[CM(k,*), CRS(i,*)] = \frac{\sum_{x=1}^{X'} W(k,x) * [CM(k,x) - CRS(i,x)]^2}{\sum_{x=1}^{X'} W(k,x)}$$

Initially the weighting factors for all variable axes for all clusters are set to

$$W(k, x) = \frac{1}{WC}$$

The clusters created are hyperellipsoidal. They shrink along the axis associated with a resource that well defines cluster membership and hence has a small within cluster variance and grow along an axis associated with a variable having large within cluster variance and hence poorly defines cluster membership.

Each iteration attempts to reassign the data points to a cluster such that the dissimilarity measure associated with it and the cluster is minimized. This measure is highly dependent upon the evaluation of the cluster centroids. Typical algorithms update the cluster centroids at the end of each iteration. The algorithm used here updates the centroid of any cluster upon the addition or removal of a data point, unless the number of cluster members is below a certain threshold, SMIN. In addition the centroids are recalculated at the termination of each iteration. The SMIN threshold ensures that there are enough members to establish the direction in which the cluster is headed. This midpass recalculation tends to speed convergence.

Before the initial data pass the $K(k)$ and $S(k, x)$ values are set to zero. Each data point, $CRS(i, *)$, is processed and if it is assigned to cluster z then

$$S(z, x) \leftarrow S(z, x) + \text{CRS}(i, x), \quad 1 \leq x \leq X'$$

$$K(z) \leftarrow K(z) + 1$$

$$\text{CM}(z, x) \leftarrow \text{CM}(z, x) \quad \text{if } K(z) \leq \text{SMIN}$$

$$\text{CM}(z, x) \leftarrow S(z, x) / K(z) \quad \text{if } K(z) > \text{SMIN}$$

At the end of the pass each $\text{CM}(k, x)$ is recalculated.

If during a pass a data point is moved to cluster z from cluster y the following calculations are made:

$$K(z) \leftarrow K(z) + 1$$

$K(y) \leftarrow K(y) - 1$; if $K(y) = 0$ then mark cluster y as empty,
set $\text{NC} \leftarrow \text{NC} - 1$ and relabel all the clusters

$$S(z, x) \leftarrow S(z, x) + \text{CRS}(i, x), \quad 1 \leq x \leq X'$$

$$S(y, x) \leftarrow S(y, x) - \text{CRS}(i, x)$$

$$\text{CM}(z, x) \leftarrow \text{CM}(z, x) \quad \text{if } K(z) \leq \text{SMIN}$$

$$\text{CM}(z, x) \leftarrow S(z, x) / K(z) \quad \text{if } K(z) > \text{SMIN}$$

$$\text{CM}(y, x) \leftarrow \text{CM}(y, x) \quad \text{if } K(y) \leq \text{SMIN}$$

$$\text{CM}(y, x) \leftarrow S(y, x) / K(y) \quad \text{if } K(y) > \text{SMIN}$$

The cluster relabelling process is such that if there were Q clusters and cluster J is to be removed then cluster $J+1$ is now called cluster J and so on through to cluster Q which is now called cluster $Q-1$.

Any decision to merge clusters is made at iteration completion. Clusters p and q are merged if

$$D(\text{CM}(p, *), \text{CM}(q, *)) < \text{DMIN}$$

If during a data pass a data point is processed such that

$$D(\text{CM}(k, *), \text{CRS}(i, *)) > \text{DMAX} \quad \text{for all } k; \quad 1 \leq k \leq \text{NC}$$

then a new cluster is created with that data point,

$\text{CRS}(i, *)$, as the centroid. The weight vector for it is set

to $1/WC$ as for the initial pass. $DMIN$ and $DMAX$ are thus the clustering parameters that control the actual formation of clusters.

At the end of each iteration the $W(k,x)$'s are calculated as

$$W(k,x) = \frac{1}{WC + VR(k,x)}$$

where $VR(k,x)$ is calculated as

$$VR(k,x) = \frac{\sum_{i=1}^{K(k)} [CRS(f(i),x) - CM(k,x)]^2}{K(k) - 1}$$

and $f(1)$ is the first value of i for which $CRS(i,*)$ is in cluster k , $f(2)$ is the next value of i and so on up to $f(K(k))$.

The algorithm terminates when an iteration results in no data points being moved.

At this time all the values are 'unscaled' and returned to their original values, including the cluster centroids. The scaling is essential in the clustering and in interpretation, as illustrated later in this chapter, yet the values required for the actual workload model are the original unscaled values.

7.1.2 Algorithm Implementation

As already mentioned, the algorithm above, except for a few changes, is basically that proposed by Agrawala. Where their scaling mechanism includes no compensation for low outliers, only for high outliers, the algorithm was extended to consider both ends. A somewhat more significant change is necessary in dealing with small clusters. Their algorithm specifies that, should a cluster exist for 3 iterations with less than 5 members, it is to be destroyed and the associated data points are to be removed from the data space. This is entirely unsatisfactory for a workload study and the algorithm was adjusted to retain all data points.

7.1.2.1 Choice of Variables

For each task over 50 different resource consumption rates are available. In certain cases each of these resources become important, however, attempting to cluster the data based on all 50 variables has minimal meaning and would be extremely expensive. This variable count consists of the resource consumption rates including 2 additional ones for each MTS I/O device: the sum of the input and output rates, and the data transferred rates. These are additional $CRS(i,x)$ to more generally describe the activity on a given MTS I/O device.

During the collection sessions studied, a number of resources were used by less than 5 percent of the tasks observed. These resources include unit record equipment,

magnetic tape, paper tape, electrostatic printer plotter, remote attached devices, and the network. Although their consumption must be considered within the final workload model their values are not used to affect clustering.

Four major resources are left to be considered:

1. CPU time;
 - problem state CPU time
 - supervisor state CPU time
2. Memory usage;
 - virtual memory size
 - working set size
 - page I/O's
3. terminal I/O; and,
4. disk file I/O.

The two CPU times must not be pooled as the problem state CPU time is a requested resource while the supervisor state CPU time is an overhead resource which is allocated to fulfill resource requests.

The 3 variables indicative of memory consumption are; virtual memory size, working set size (as measured by the supervisor), and page I/O rate. Clustering was attempted using various combinations of these 3 variables and the final decision was to retain all 3. Working set size and virtual memory size are each measures of requested resources while the paging I/O rate is an overhead function that describes the response of the system to the request of information to be transferred from virtual memory to the

working set. The intricate way in which these 3 variables interact means that the only way in which a true workload description of their usage can be obtained is to consider all 3 as essential variables.

For terminal and file system disk activity 7 variables are available: number of initializations, number of input operations, total length of data transferred by all input operations, number of output operations, total length of data transferred by all output operations, total number of input and output operations, and total length of data transferred. Once more, remembering that the desired result is a small number of broad clusters, it is more logical to consider the input and output operations together. The actual number of I/O operations initiated is a good measure of the activity. Hence, the variable chosen to describe terminal and file system activity is the rate of input and output operations for each of these devices.

Thus, out of all X variables available 7 are selected as the most important ones in characterizing task demands, hence X' is 7. Clearly the values of the other variables still exist for all tasks but are not used in the clustering.

7.1.2.2 Choice of Initial Cluster Centroids

To start the clustering process a certain number of clusters and their centroid values need to be specified. This problem of the initial number and their locations has

to be solved. For example, starting with a single cluster involves many iterations in which the naturally existing task groups are separated out into new clusters, whereas starting with a very large number of clusters again involves many iterations to merge the artificially induced groups. Test runs indicated that 5 distinct clusters emerge and for this reason 5 initial clusters are used. To determine which centroids these initial clusters should have, two approaches were tested. One approach used 5 randomly selected data points and the other approach used 5 points which spanned the 7 dimensional hypercube. It should be noted that the choice of 5 random points may result in all 5 of them ultimately belonging to one cluster and hence defeating the initial choice of 5 clusters, increasing the number of iterations. Both techniques were applied to a number of segments. This is a good test to ensure that natural clusters do exist, since they will evolve from the data regardless of the initial cluster number and their centroids. Although the cluster centroids moved somewhat, and the members in each cluster adjusted slightly the same basic 5 cluster structure existed for both cases. The approach finally used is a set of 5 equally spaced centroids spanning the hypercube diagonal. This provides faster convergence since, regardless of how the initial centroids are chosen, the resultant clusters generally span the hypercube.

7.1.2.3 Clustering Software

Two programs are used to perform the clustering analysis. The first program, the segment extractor, written in assembler, converts the task by task resource consumption rate data to a segment by segment format. For each segment, the identification data and resource consumption rate data is extracted from all workload data records that measure activity during that segment. The values are also converted from their internal representation to a character format readable by the clustering program. The two additional MTS I/O counts of total inputs and outputs, and total length of data transferred are also inserted.

As the workload model is built upon the resource consumption rates of each cluster type in association with the number of cluster members present during any model interval, the actual times of the model intervals must also be prepared for input to the clustering program which will ultimately produce the workload model. Hence, the time associated with the end of each model interval is also recorded. While processing these times, this program furthermore performs the analysis of the snapshot exposures for the collection session producing the minimum, maximum, mean, and standard deviation of exposure time, as presented in Chapter 6.

The input to the program consists of:

1. the output file 1. from the workload conversion program
 - providing the resource consumption rates for each

task; and

2. the QQ file - providing the data to calculate the model interval.

The output from the program consists of:

1. the resource consumption rate data split and reordered segment by segment and the model interval times; and,
2. the analysis of the snapshot exposure times.

The second program, written in ALGOLW, uses the segment by segment data from the output of the first program in the clustering analysis algorithm described in all of Section 7.1 above. This program allows the user to specify which of the 50 or so X variables should be used to determine the clusters, whether or not non-MTS tasks should be included in the clustering, and finally the clustering parameters. The final output is the workload model giving for each segment: the number of clusters within the segment and specifics of each cluster, including arrivals and departures. For each resource consumption variable the minimum, maximum, mean, and standard deviation within the cluster are given regardless of whether or not the variable was involved in determining the clusters. All these values are in terms of the unscaled values $CR(i,x)$, as once the clusters are established the values are converted back to their original unscaled form.

The input to the program consist of:

1. a specification of the clustering parameters, SMIN, WC, DMIN, and DMAX, as well as the variables defining the

cluster space; and,

2. the output file 1. from the segment extractor.

The output from the program consists of:

1. detailed data on the formation of the clusters along with the final result giving the clustering variables in both their unscaled and scaled format as well as the non-clustering variables - supplying minimum, maximum, mean, and standard deviation for all variables; and,
2. the workload model as described in Section 7.2.

7.1.3 Clustering Results

The clustering is used to determine task groups within each segment. However, the analysis showed that a consistent set of 5 clusters exists across all segments of all collection sessions. Occasionally an exceptional task will cause an additional cluster to be formed for it alone, and in quiet periods one of the clusters might be missing altogether.

The clustering algorithm was attempted with a variety of cluster variables and cluster parameters. The choice of the final set of variables is explained above. The values of the cluster parameters were determined by analyzing various test runs. The effect of various values of WC and SMIN were found to be minimal and hence were set to the values used by Agrawala, .1 and 5, respectively. DMIN and DMAX are much more important. They are set to .5 and 3, respectively. When a value of less than 3 is used for DMAX then at least an

additional 5 clusters appear, each with a single member. This destroys the general broad classification required. Hence, the lowest value of DMAX, 3, which generates a broad set of clusters was decided to be the optimum value.

The convergence of the algorithm is rapid, generally converging in less than 10 iterations, and always converging in less than 20 iterations.

As mentioned above, 5 general clusters evolved from within the data. A series of Kiviat graphs are provided in Appendix B which illustrate these clusters. A series of segments are illustrated for the 10 a.m. run from May 24th followed by the first segment of each of the subsequent runs. The first of these figures is reproduced as Figure 9 for easy reference. Each figure illustrates the 5 clusters formed during the segment, with each Kiviat graph representing one of the clusters. The clusters A through E appear in the same order as the algorithm generated them based on the initial 5 seed centroids. The values plotted are the scaled values for the centroids (bold line) and the within cluster standard deviation (light line) for each clustering variable. As these are scaled values each variable axis runs from 0 to 10. A low standard deviation on a given resource axis implies that that resource distinguished well between members of that cluster versus non-members. Therefore the standard deviations can be used to recognize the resources that determined a given cluster.

For any segment, 5 basic clusters exist. Two of the

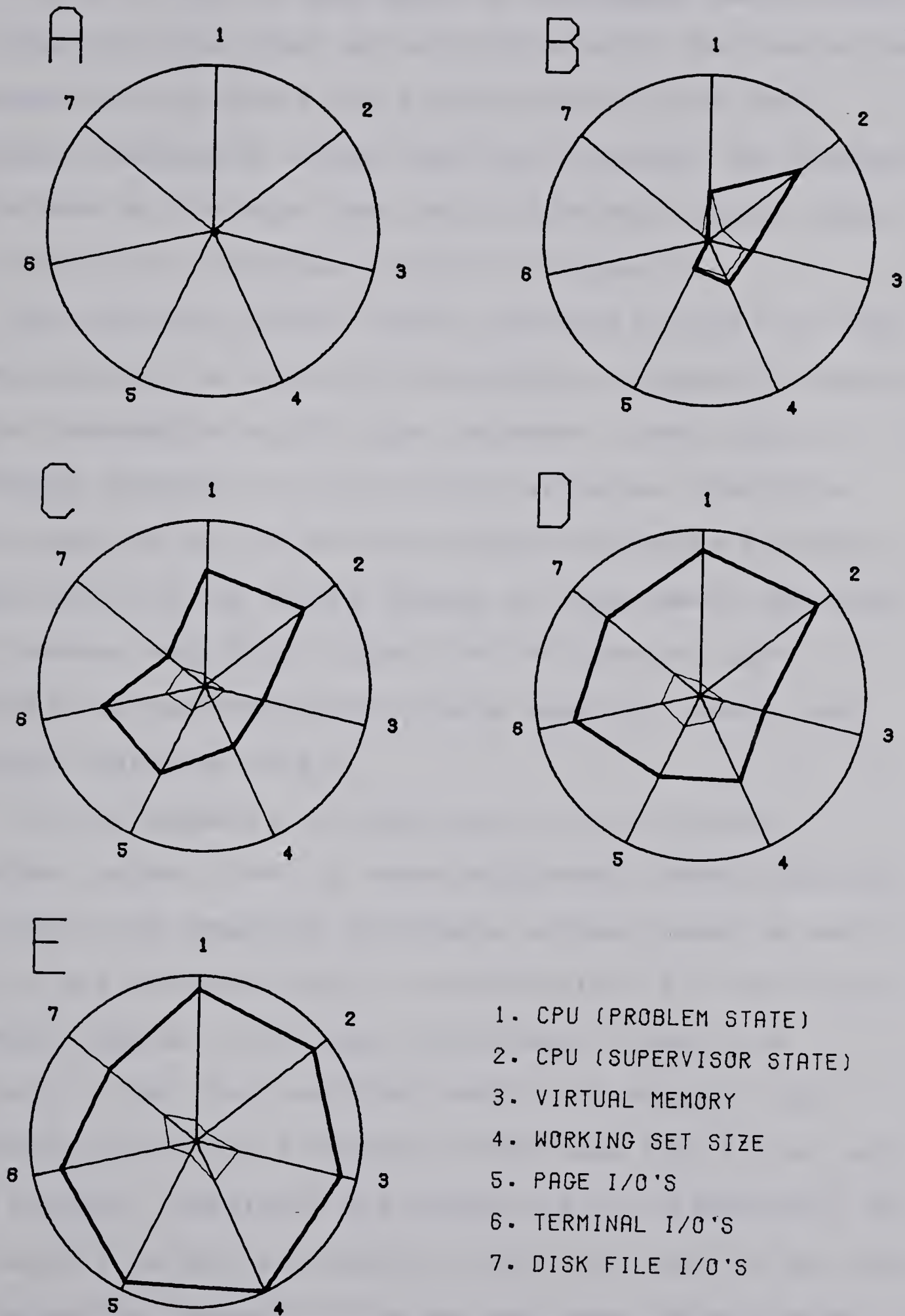


FIGURE 9 - Task Groups - May 24, 10 a.m. - Segment 1

clusters, which include all of the non-MTS tasks, show little or no usage of disk files or terminals, one of these two clusters often shows no activity at all. The lack of use of terminals and disk files are the factors that best determine membership within these two clusters, the standard deviations on the other axes being relatively large. These two clusters are clusters A and B in Figure 9.

The remaining three clusters generally group the other tasks primarily on their CPU consumption. Generally speaking as the consumption of CPU time increases across these 3 remaining clusters so do the other variables. Disk file I/O's often do not follow this general principle and show a higher usage in the middle cluster of this group. The tasks that consume very large amounts of CPU time per unit interval are necessarily performing less I/O. Hence, the observed reduction in I/O.

In some segments, an additional 1 to 3 clusters sometimes appear. Each of these additional clusters has but one member and generally represents extreme usage in one or more of the variables (i.e. a scaled value of > 11). It is critical however, that these exceptional clusters be included in the final workload model. For example, this occurred for the first segment of the June 7th, 2 p.m. run. This cluster, containing one member, is not illustrated. It was unusual in that its scaled VM size was over 19, and both its supervisor state CPU time and and page I/O's exceeded 10.

Extreme slow periods such as those illustrated by the 8 p.m. run on June 28th, and the 1 a.m. run on June 27th are basically the same with minor differences. During both of these sessions very little paging was done. In both Figures 18 and 20 clusters E and A correspond to the first two clusters discussed above. Clusters C and D correspond to the second set of clusters described above. Often in periods of low activity this group of users formed only two clusters rather than the 3 observed elsewhere, which is surely due to the general lack of activity at these times. One additional cluster which does not fall into the general pattern described above also appears in each of Figure 18 and 20, the cluster labelled B. Only a small portion of the jobs required any paging during these periods, and the amount of paging they requested was small. However, because they requested any at all the scaling mechanism indicated they required the most paging. Because their small amount of paging was converted to a large scaled value these tasks formed their own cluster. Hence, the general clusters still occur within periods of lower usage, with occasional exceptions.

Included at the end of Appendix B are six tables which supplement Figures 11 through 20. Table 4 gives the number of tasks in each cluster and the percentage they form of all the tasks observed. Tables 5 and 6 give the minimum, maximum, and average number of tasks active during any model interval for each of the segments illustrated. The values

are given for MTS and non-MTS tasks. Tables 7, 8, and 9 supply the actual unscaled resource consumption rates for the variables forming the axes of the Kiviat graphs. The values show how widely spread the actual values can be for two different segments with very similar scaled clusters. For example, cluster E in Figures 11 and 12 indicate approximately the small scaled value of CPU problem state time, however the actual values as given in Table 7, show that the actual consumption rate was over twice as high in the second segment.

7.2 Workload Model

The workload model suitable as input to a system model can now be formulated. The workload model specifies the workload characterization for each segment. The segment format consists of a set of K resource consumption rate vectors where K clusters have been determined within the segment. Such a segment format is given in Figure 10. The vector consists basically of the resource consumption rate vector with the addition of the two MTS I/O rates. The final workload model therefore consists of a series of workload models for each segment of the collection session.

As discussed in Chapter 4 this model includes several features and techniques heretofore not considered in other studies. These changes produce a workload model that has not significantly increased in complexity yet is more powerful.

***** Once per Segment*****

- The endpoints of each of the S+2 model intervals
TMIN(0), IMID(0,1), ... IMID(S,S+1), TMIN(S+1).
- The number of clusters in the segment.

*****Data Section that is retained for each cluster*****

- The number of MTS and non-MTS tasks in this cluster
active during each of the S+2 model intervals.

For each of the following the values given are the minimum,
maximum, average, and standard deviation of that resource
across all tasks in the cluster.

- Virtual memory size.
 - Working set size.
 - Problem state CPU consumed per second in 1000ths of a
microsecond.
 - Supervisor state CPU consumed per second in 1000ths of a
microsecond.
 - Page I/O's done per second times 1000.
 - HASP pages printed per second times 1000.
- For each of unit record devices, magnetic tape drives,
terminals, disk files, paper tape, electrostatic printer
plotter, remote attached devices, and network devices
the following 7 fields appear, where each one is
retained internally as times 1000:

=====

- Initializations per second.
- Input operations per second.
- Total data length involved with input operations per
second.
- Output operations per second.
- Total data length involved with output operations per
second.
- Sum of input and output operations per second.
- Total data length involved in all input and output
operations per second.

=====

- The total number of think times measured in the segment.
- The total time due to think times in the segment
measured in 1/122nds of a second.
- The total number of response times measured in the
segment.
- The total time due to response times measured in the
segment measured in 1/122nds of a second.

FIGURE 10 - Workload Model - Segment Format

8. Summary, Conclusions, and Future Research

The workload model designed and implemented is based on Bard's approach (BARD76a, BARD76b, BARD77, BARD78) to the problem and uses Agrawala's clustering technique (AGRA76, AGRA77a, AGRA77b, AGRA78) to determine natural groupings of task demands. The final model is considered to be superior to those on which it is based because the workload models retains much more information (see Section 4.3) without significantly increasing the size or complexity of it.

The model as it stands is suitable to form the workload input to a system model which is designed to deal with hardware and software resource changes (see Section 4.1). With further analysis it is also suitable to generate a synthetic jobstream.

8.1 Workload Model

The highlights and improvements incorporated in the present workload model were mentioned in general terms in Chapter 4; the specific points are summarized below.

- As in previous studies it was determined that the volume of data to be processed requires reduction. However, this reduction must bring out the inherent characteristics of the real workload instead of hiding them. Hence, task groups are determined from within the data where each group is parameterized by the average of its members. Instead of manual classification a general

clustering algorithm is employed.

- A one to three hour measurement session was found to be necessary. Previous studies reduced the data from each session to one set of values, averaging out all peaks in the data. The current study retained peak and valley information by defining segments within the total period.
- The general technique of measuring total resource consumption is replaced by the use of rates and precise measurement of the arrival and departure of each task. The change here is much more than simply replacing totals by rates, it is a detailed analysis of the lifetime of the task in conjunction with its consumption. Although such an analysis complicates the final workload model very little it retains an incredible amount of additional information.
- Finally by measuring consumption from above the lowest software level the workload model can be used as input to a system model designed to study hardware resource changes and many software resource changes.

Ferrari's (FERR78) general characteristics of workload models stated in Chapter 2 are now discussed in terms of the present workload model.

- Representativeness:

The workload model is built from observed workload data which is validated and the resultant task groups are allowed to form naturally. The inclusion of segments in

the entire collection session ensures the retention of peak and valley information. All of these features ensure a representative workload model.

- **Reproducibility:**

Once constructed the workload model will always represent the same distribution of resource demands.

- **Flexibility:**

The workload model is completely flexible given the current command structure of MTS and the current hardware resources. As additional hardware resources are added to the system, modifications to the software tools would be required to include them. Additionally a workload model can be built for any requested time period.

- **Simplicity of construction:**

The software tools developed make the creation of a workload model extremely simple and completely automatic. However, the development effort and cost of the detailed monitoring and analysis must not be overlooked in the construction of the resultant compact model.

- **Compactness:**

The techniques employed reduce massive amounts of data to a very compact model which is primarily intended as input to a detailed system model. As it stands it is still too detailed to be used directly to generate a synthetic jobstream but further analysis can reduce it

to the level required.

- Usage costs:

The costs of using it are the costs of running the system model or the synthetic jobstream.

- System independence:

It is entirely dependent on MTS but is built to be unaffected by various low level changes in the MTS software, as for example changes due to a new file system or a new spooling system. Additionally, some hardware changes do not affect the construction of the workload model at all, for example the upgrade to the AMDAHL V/7 required no changes. Generally changes would be required for the addition of new hardware but not for upgrades. The supervisor CPU time is dependent upon the structure of the supervisor and may require adjustment, if it is used as input to a system model based on a new supervisor. However, supervisor CPU time is likely to be a resource built into the system model rather than accepted as a workload input parameter.

- Compatibility:

It is compatible directly for the system model application and with modification can be used to generate a synthetic jobstream.

Such a model allows for very detailed studies of MTS. In contrast a system such as that described by Buzen (BUZE78) appears very useful, and much less specialized, but once the specifics of a particular application are

considered it becomes doubtful if such a general purpose tool could ever perform well. Buzen's workload model is constructed of various user types, such as batch and terminal, with the number of milliseconds of servicing required from each device each time a member of that user type is given the processor. Such a workload model is difficult to build in addition to being somewhat unrealistic. The current study proposes a more specialized workload-system model, but one that is accurate for its intended purpose.

8.2 Future Applications

The two intended applications provide a very powerful set of performance tools for MTS. Each application is a significant further research study.

8.2.1 System Model

The system model has been described in Section 4.1. It is a general purpose tool that could be used to study the effect of almost any change to the hardware or software resources on the performance of the system.

The workload model interfaces with the system model by the segment by segment data composed of details on each cluster in the segment as shown in Chapter 7. The system model would process the workload data, a model interval at a time. Each model interval contains the number of members for each type of cluster present. If the number changes from one

model interval to the next, the difference is processed as either the arrival or the departure of tasks. Each segment of workload data specifies a new set of clusters, hence, at segment end the task profile associated with a given cluster changes, all tasks from the previous segment depart, and all new tasks arrive. Although the means can be used, the standard deviations, minimum, and maximum of each rate for each task group is also given, so a distribution based on this data could replace the mean for an even more accurate representation.

Careful consideration would be required with regard to which non-MTS tasks to include and which to exclude for the workload model.

8.2.2 Synthetic Jobstream

The workload model available is sufficient to be used in the construction of a synthetic jobstream. However it requires reduction before it can be used to generate such a synthetic jobstream.

Although detailed methods exist for calibrating synthetic jobstreams the actual success of a synthetic jobstream lies in the representativeness of the synthetic job or script used. It is not a precise methodology, but rather an attempt to define a task that will consume resources at a specified rate, or preferably to construct a program which will generate the job or script according to specified rates.

Once the stream of jobs is defined a driver is used to run the system as if this was a production stream. Such a driver already exists for MTS.

Therefore the steps required to derive such a synthetic jobstream, which is but another type of workload model, are twofold. Firstly, the data provided in the current workload model must be reduced. The set of task group definitions at each segment must be replaced by a set that can be used to represent activity for the entire session. This reduction must be performed carefully, with close attention to workload peaks. A detailed study of this portion alone could be extensive. Secondly, a synthetic job generator would have to be developed. The specifications for each task group would be supplied to the generator and a script would be produced. The number of these scripts started would be equal to the number of tasks in the task group.

8.3 Future Research

In the area of the workload model itself, additional significant studies could be done based on the work presented here. Now that the software tools have been developed, the process of building a workload model for any required period is straightforward, allowing any of a number of studies to be undertaken.

The structure and flexibility of the workload model is suited to studies as basic as that of studying the effect of different charging schemes on user demands. Comparisons

could be made on the demands the users make as the costs of the various resources change. This would be a long term study as changes in the short run would force all users to do all their work during the period with the lowest charges.

The clusters lend themselves to a detailed time series analysis. This would involve determining ways of applying time series analysis to clusters. Such a technique could be used to calculate projected workload on which hardware planning could be based.

An additional study could centre more closely on the clustering parameters and the relationship between sets of clusters created from the same data but using different clustering parameters. A detailed analysis of the actual data loss incurred by reducing to a given number of clusters could follow.

The idea of true system independence has not been accomplished in this study, but perhaps similar techniques could be applied to the truly system independent measures mentioned in Chapter 2 such as the virtual memory required for a table look-up implementation.

A number of the standard problems with workload modelling have been overcome in this study. Further work and extensions based on similar techniques may succeed in overcoming additional problems.

Bibliography

[Note: Some references are included that are not explicitly referred to in the text. They are relevant to the study of workload modelling and are included for the sake of completeness.]

- AGRA76 Agrawala, A.K., Mohr, J.M., Bryant, R.M., "An Approach to the Workload Characterization Problem", Computer, June 1976, pp. 18-32.

The paper discusses three types of workload models: a model characterized by the resource requests and by the source of the requests, a model which considers additionally the time at which the request is placed, and a model which considers, in addition to the above three factors, the location from which the resource is placed. The first model employs clustering analysis. The other two are stochastic models using time series and Markovian models.

- AGRA77a Agrawala, A.K., Mohr, J.M., "Some Results on the Clustering Approach to Workload Modelling", University of Maryland, Department of Computer Science, Technical Report No. 521, April 1977.

The paper discusses the application of clustering techniques to workload characterization. The types of workload variables considered are vectors of resource demands, percentages of program types, and vectors of the relative usage of each resource. Natural groupings are observed and the clustering algorithm is shown to be stable.

- AGRA77b Agrawala, A.K., Mohr, J.M., "The Relationship Between the Pattern Recognition Problem and the Workload Characterization Problem", Proc. of ACM SIGMETRICS/CMG VIII Conference on Computer Performance: Modelling, Measurement, and Management, November 1977, pp. 131-139.

The authors present the notion that so far no models exist to formulate the behaviour of a user. Rather, workload models are built based on the behaviour observed from data collected from the system. The volume of these data can be reduced by

recognizing patterns or clusters. Techniques exist in pattern recognition theory allowing models to be formulated based on the pattern environment. This technique can be carried over to aid in the workload characterization problem. Care must be taken as there are basic differences between the pattern recognition and clustering application. Primarily in the workload application, there is no a priori set of classes as assumed in pattern recognition theory. The method used for determining the set of features that best describe the clusters is based upon the error or misclassification of data points that occurs when a feature is removed.

AGRA78 Agrawala, A.K., Mohr, J.M., "Predicting the Workload of a Computer System", AFIPS Proc. NCC, 1978, pp. 465-471.

The paper addresses workload prediction and shows that accurate results can be obtained by determining the natural groupings, or clusters, in the current workload and then applying statistical forecasting techniques to these clusters. The clusters are based on the system resource usage levels of each task. The types of changes considered are random stochastic variation, monotonic, oscillatory, discontinuous, and composite changes. In general, the overall workload will change and the clusters will have moved but the type of jobs in each cluster will remain the same.

ANDE72 Anderson, H.A., Sargeant, R.G., "A Statistical Evaluation of an Experimental Interactive Computing System", Statistical Computer Performance Evaluation, Ed. W. Freiburger, Academic Press, New York, 1972, pp. 73-98.

A statistical evaluation of the scheduler for IBM APL/360 on an IBM 360/50 is based upon data collected during normal production periods. An attempt is made to determine the statistical properties of the workload and performance variables. Regression analysis is used to develop models for the performance variables based upon the workload and system parameters.

ANDE73 Anderberg, M., Clustering Analysis for Applications, Academic Press, New York, 1973.

A comprehensive introductory presentation of clustering techniques, with emphasis on application.

- ARBU66 Arbuckle, R.A., "Computer Analysis and Thruput Evaluation", Computers and Automation, Vol. 15, No. 1, January 1966, pp. 12-15.

In discussing evaluation of computer systems the concern that all factors must be considered is presented. Throughput is presented as a good measure and suggests that a hardware monitor measuring throughput can yield information on CPU and channel usage to locate bottlenecks.

- BARB75 Barber, E.O., Asphjell, A., Dispen, A., "Benchmark Construction", ACM SIGMETRICS Performance Evaluation Review, Vol. 4, No. 4, October 1975, pp. 3-14.

The paper dicusses work done at the University of Trondheim in Norway to aid in testing the performance of new machines. A program for their UNIVAC 1108 EXEC 8 processes accounting data and produces a synthetic benchmark stream. The stream consists of Fortran and Algol jobs to increase portability.

- BARD76a Bard, Y., "A Characterization of VM/370 Workloads", IBM Cambridge Scientific Center Technical Report No. G320-2111, April 1976.

The paper presents the details on a workload model built to drive an analytic model of VM/370. The model is used as a tool to choose an optimal hardware configuration for a given workload. The workload model is based upon task resource demands. The model is constructed of average resource consumption for a number of different classes of users. The classes of users are very dependent upon the VM/370 scheduler. The portability question is also addressed and dealt with in the context of VM/370.

- BARD76b Bard, Y., "A Characterization of VM/370 Workloads", Modelling and Performance of Evaluation of Computer Systems, Ed. H. Beilner and E. Gelenbe, North-Holland, Amsterdam, October 1976, pp. 35-55.

This is a published version of the technical report (BARD76a) with only minor corrections and changes.

- BARD77 Bard, Y., "An Analytic Model of the VM/370 System", IBM Cambridge Scientific Center Technical Report No. 2121, May 1977.

The paper describes a model for predicting system performance for a given workload under a specified hardware configuration for VM/370. A transaction oriented workload is developed from data collected during a two hour interval. It is resource demand based and adjustments are made to compensate if the test machine differs from the source machine. User classes are determined manually and average resource demands calculated. The workload data, in conjunction with the hardware configuration, is presented to an analytic model which provides information on the resource utilization and system response. The flow of jobs through the system is modelled using Little's formula for either FIFO or fair share scheduling. The overall model consists of the flow model in conjunction with a closed queuing network that describes the behaviour while the job is being actively serviced.

- BARD78 Bard, Y., "An Analytic Model of the VM/370 System", IBM Journal of Research and Development, Vol. 22, No. 5, September 1978, pp. 498-508.

This is a published version of the technical report (BARD77) with only minor corrections and changes.

- BOIE74 Boies, S.J., "User Behaviour on an Interactive Computer System", IBM Systems Journal, Vol. 13, No. 1, 1974, pp. 2-18.

An ongoing project at the IBM T. J. Watson Research Center, to understand the factors that determine the level of human performance in interactive computer systems, is discussed. Issues considered include language processors, command usage, and user response time. The results are collected using SPIE, System Performance Internal Evaluator.

- BUCH69 Buchholz, W., "A Synthetic Job for Measuring

System Performance" IBM Systems Journal, Vol. 8, No. 4, 1969, pp.309-318.

A synthetic job is proposed that is suitable for comparing the performance of differing systems with minimal conversion effort required. It is a simplified file maintenance program written in PL/I.

- BUCK69 Buckley, F.J., "Estimating the Timing of Workload on ADP Systems: An Evaluation of Methods Used", Computers and Automation, Vol. 19, No. 2, February 1969, pp. 40-42.

Various methods for estimating the amount of time required for a specific system to process a given workload are discussed. Among these are system timings based on internal timings and internal bulk characteristics of the workload. Also included are simulation using more specific workloads than possible in the previous methods, and, finally, benchmarking.

- BUZE78 Buzen, J.P., Goldberg, R.P., Langer, A.M., Lentz, E., Schwenk, H.S., Sheetz, D.A., Shum, A., "BEST/1 Design of a Tool for Computer System Capacity Planning", AFIPS Proc. NCC, 1978 pp. 447-455.

BEST/1, a modelling tool designed to answer performance questions, is discussed. They claim that it is capable of calculating the performance of almost any system and is based internally on the theory of multiple class queuing networks. The user need deal only with a high level language through which the system's software, hardware, and workload are described. The central importance of the workload question is recognized. Workloads presented to the model can be in the form of time sharing data, transaction rate data, or batch loads.

- CALL75 Callaway, P.H., "Performance Measurement Tools for VM/370", IBM Systems Journal, Vol. 14, No. 2, 1975, pp. 134-160.

The tools available to the users as well as the implementors of VM/370 for monitoring performance measurement are presented. Users can obtain load level indicators allowing them to better schedule their work and resource tradeoffs. Analysts can monitor the system in real time or

use accumulated data.

- CHAN77 Chanson, S.T., Bishop, C.G., "Simulation Study of Adaptive Scheduling Policies on Interactive Computer Studies", ACM SIGMETRICS Performance Evaluation Review, Vol. 6, No. 3, Summer 1977, pp. 33-39.

The paper discusses a simulation model used to test various scheduling algorithms. The model is a simplified version of MTS at the University of British Columbia. The workload is initially divided into six user classes and then jobs for each user class are constructed. The input from these jobs to the simulator is in the form of an I/O request with specifics supplied, or a terminal request with the CPU usage supplied. It is not clear how the user groups were determined.

- CHEN69 Cheng, P.S., "Trace Driven System Modeling", IBM Systems Journal, Vol. 8, No. 4, 1969, pp. 280-289.

The paper describes a simulation model driven by a resource allocation trace derived from data collected while running a series of controlled jobs on the system. The author proposes this as a way to ease the task of producing a system model as the trace data itself provides detailed information such that the simulator itself need not be as extensive.

- FANG76 Fangmeyer, H., Gloden, R., Larisse, J., "An Automatic Clustering Technique Applied to Workload Analysis and System Tuning", Modelling and Performance of Computer Systems, Ed. H. Beilner and E. Gelenke, North-Holland, Amsterdam, October 1976, pp. 427-433.

Basic clustering techniques are applied to two accounting variables from a month's accounting data from an IBM 370/165 OS system. The authors are interested in clustering only a small number of variables at a time, using the results to determine the relationship between the usage of the chosen variables.

- FERR72 Ferrari, D., "Workload Characterization and Selection in Computer Performance Measurement", Computer, July/August 1972, pp.18-24.

The author addresses a number of alternatives to the workload characterization problem and discusses the costs and tradeoffs involved. A particular concern is deriving a characterization that is representative of the real workload.

FERR78 Ferrari, D., Computer Systems Performance Evaluation, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

This book is an in-depth study in the entire area of performance evaluation. The author discusses the various types of measurement, simulation, and analytic tools available, in conjunction with their appropriate applications. Workload modelling is presented along with each of these techniques. In addition, an entire chapter is devoted to workload modelling. Many ideas are presented in detail while significant current literature is at least touched upon.

FLYN74 Flynn, M.J., "Trends and Problems in Computer Organizations", Information Processing 74 (Proc. IFIP Congress 74), North Holland, Amsterdam, August 1974, pp. 3-10.

The author is concerned with current and future architectures and the need of the designers to be familiar with the use and efficiency of the software and of the hardware. The paper looks at the efficiency of the IBM 7090 and 360 architectures in conjunction with the Gibson mix, described in (GIBS70), and with the data collected by Winder (WIND73).

FOGE72 Fogel, M., Winograd, J., "EINSTEIN: An Internal Driver in a Time-Sharing Environment", ACM SIGOPS Operating Systems Review, October 1972, pp. 6-14.

The paper describes a program written to allow a reproducible load to be run on a UNIVAC Series 70, under VMOS. The load can be a mixture of batch and time sharing applications and a task is presented as either a terminal script or as a specified series of resource demands.

GIBS70 Gibson, J.C., "The Gibson Mix", IBM Technical Report No. 00.2043, June 1970.

A particular instruction mix is defined. The paper was unavailable but is used as a base reference in other publications such as those by Ferrari (FERR78) and Svobodova (SVOB76).

- GOMA76 Gomaa H., "A Modelling Approach to the Evaluation of Computer System Performance", Modelling and Performance of Computer Systems, Ed. H. Beilner and E. Gelenbe, North-Holland, Amsterdam, October 1976, pp. 171-199.

A system model for the CDC 6000 Kronos system is described. The system model employs simulation and regression techniques and accepts as input a regression workload model. The workload model expresses the elapsed time of batch jobs in terms of the job's resource demands. The workload model is composed only of those jobs that perform well under the regression workload model constructed, the remainder being discarded.

- HART75 Hartigan, J.A., Clustering Algorithms, John Wiley & Sons, New York, 1975.

A detailed theoretical study of clustering, with some applications inserted.

- HELL72 Hellerman, L., "A Measure of Computational Work", IEEE Trans. on Computers, Vol. 21, No. 5, May 1972, pp. 439-446.

The author is concerned with a measure of the computational work required by a given data process that is independent of a particular machine and the associated instruction set. A data process is defined as a set of inputs, a set of outputs, and an assignment of a unique output for each input. The proposed notion of computational work is based upon this definition and measured in terms of the information that must be stored in memory for its table lookup implementation.

- HILL66 Hillegass, J.R., "Standardized Benchmark Problems Measure Computer Performance", Computers and Automation, Vol. 15, No. 1, January 1966, pp. 16-19.

The paper discusses a system of standardized benchmarks used to compare systems. Given the manufacturers' timings the performance of the

system on this application is calculated allowing for comparisons. This calculation produces a result very near to that produced when the benchmark stream is actually run.

- HUNT71 Hunt, E., Diehr, G., Garnatz, D., "Who are the Users? - An Analysis of Computer Use in a University Computer Center", AFIPS Proc. SJCC, 1971, pp. 231-238.

A description of the methods by which a categorization of the users of a CDC 6400 system at the University of Washington was attempted is presented. For resource usage on a per task basis, the authors investigate the statistical properties of the variables, including the correlations between measures. Cluster analysis succeeds in defining the user groups in terms of their resource demands.

- JOSL66 Joslin, E.O., Aiken, J.J., "The Validity of Basing Computer Selection on Benchmark Results", Computers and Automation, Vol. 15, No. 1, January 1966, pp. 22-23.

The use of benchmarks in selecting computer systems is considered valid given that the benchmarks have been constructed properly. Different machines perform better on different portions of the workload. It is important to know the relative weights of these functions in the real workload.

- KERN73 Kernighan, B.W., Hamilton, P.A., "Synthetically Generated Performance Test Loads for Operating Systems", Proc. First Annual SIGME Symposium on Measurement and Evaluation, February 1973, pp. 121-126.

The paper describes the development of a synthetic benchmark tool used to study a dual processor Honeywell 6070 running GECOS III. A generator converts accounting information into a synthetic jobstream by use of a parameterized synthetic job. They find that relatively little detail is required and that the system performance measures are insensitive to the internal structure of the jobs composing the workload.

- KLEI75 Klein, R., "Software Tools to Construct Synthetic

Benchmark Jobstreams", Share Inc. Computer Measurement and Evaluation Selected Papers from the Share Project, Vol. 3, 1975, pp. 294-300.

The paper discussed the software tools developed at Bell Telephone Laboratories to construct a representative synthetic batch jobstream based on resource utilization. The tools include a synthetic job program, calibration programs, and a job generator to turn usage statistics into a runnable synthetic jobstream. The synthetic job is based upon that described by Bucholz (BUCH69) with extensions to the I/O area. This is a companion paper to the paper by Ritacco (RITA75).

KNIG66 Knight, K.E., "Changes in Compiler Performance - A Historical Review", Datamation, Vol. 12, No. 9, September 1966, pp. 40-54.

The author discusses the performance changes encountered during the industry's first 20 years. He studied workload by analyzing the frequency of instruction types in several programs. He then derived a formula for the performance of a computer for scientific and commercial applications based on the study. For 225 machines, their performance was calculated using their internal timings and the derived formula.

KUMA78 Kumar, L.B., Davidson, E.S., "Performance Evaluation of Highly Concurrent Computers by Deterministic Simulation", Communications of the ACM, Vol. 21, No. 11, November 1978, pp.904-913.

A study to investigate the performance of alternative hardware configurations of the CPU-memory subsystem of the IBM 360/91 is discussed. A deterministic simulator is used. The data used to drive the simulator is referred to as a control stream and is generated in one of two ways. Only one program is considered at a time and the control stream is either based upon instruction frequency data or trace data collected while running the program.

LAND75 Landwehr, C.E., "Usage Statistics for MTS", ACM SIGMETRICS Performance Evaluation Review, Vol. 4, No. 2, April 1975, pp. 13-23.

The author discusses usage statistics for MTS at the University of Michigan collected during November 1973, while running on an IBM 360/67. Also included is information from October 1974 showing the increased load. Included are the number of batch jobs and terminal jobs by hour of the day, CPU utilization, and batch jobs awaiting execution. Consideration is given to the resources consumed by the overhead tasks.

- LASS73 Lassetter, G.L., Lo, T., Chandy, K.M., Browne, J.C., "Statistical and Pattern Based Models for CPU Burst Prediction", Proc. Computer Science and Statistics: 7th Annual Symposium on the Interface, October 1973, pp. 123-129.

Concerned with analyzing and predicting CPU burst patterns, a Markov process is used to model CPU burst generation. CPU burst patterns for 400 jobs are taken from trace tapes produced on the UT-2D operating system for the CDC 6600 and analyzed to produce the model. The conclusion is that a scheduler that can simply recognize whether a current burst is long or short is 90 percent as effective, performance wise, as a scheduler that can actually predict CPU bursts in advance.

- LUCA71 Lucas, H.C. Jr., "Performance Evaluation and Monitoring", Computing Surveys, Vol. 3, No. 3, September 1971, pp. 79-85.

This is an overview paper that considers the purposes of performance evaluation to be selection, evaluation, and projection. The methods considered as tools are internal timings, instruction mixes, benchmarks, synthetic programs, simulation, and analytic models. In addition, a kernel program is considered as a type of benchmark that utilizes only the CPU, and not I/O, and allows for comparisons of manufacturers' timings with the actual timings.

- MAMR77 Mamrak, S.A., Amer, P.D., "A Feature Selection Tool for Workload Characterization", Proc. of SIGMETRICS/CMG VIII Conference on Computer Performance: Modelling, Measurement, and Management, November 1977, pp. 131-139.

The paper addresses the problem of selecting a subset of the total workload variables. The variables used are resource request vectors. The

algorithm works with the clustering techniques described by Agrawala (AGRA76, AGRA77a, AGRA77b, AGRA78). One subset of variables is considered more effective than another in determining cluster membership if the estimated probability of error in describing cluster membership is less for the first. The final subset ultimately depends on the function of the subset and how closely it describes the actual workload. If the generated workload is to be used to study a particular area of the system, then the variables directly related to the area under study should be included.

MEAD78 Mead, R.L., Schwetman, H.D., "Job Scripts - A Workload Description Based on Event Data", AFIPS Proc. NCC, 1978, pp. 457-464.

The generation of the job scripts used as input to the trace driven simulation model of the CDC 6500 at Purdue University is discussed. A monitor collects event data while the system is running. This data is then manipulated to remove load dependent activity. The method is considered complete and as requiring few overriding assumptions. It is, however, expensive and bulky. The volume of data can make it difficult to form any overall conclusions.

MORG73 Morgan, D.E., Campbell, J.A., "An Answer to a User's Plea?", Proc. First Annual SIGME Symposium on Measurement and Evaluation, February 1973, pp. 112-120.

The proposal given is that synthetic jobstreams are the most cost-effective technique for choosing the best system or service bureau. Synthetic jobs built at the resource demand level and the service demand level are considered, with the latter being the more portable.

OLIV74 Oliver, P., Baud, G., Cook, M., Johnson, A., Hoyt, P., "An Experiment in the Use of Synthetic Programs for System Benchmarks", AFIPS Proc. NCC, 1974, pp. 431-438.

The paper describes an experiment in using synthetic programs to attempt to overcome the limitations imposed by standard benchmark techniques. The authors find that a small number of synthetic jobs can be used to represent a varied workload, via parameterization. Individual

modules prove useful for isolating particular resource usage.

- ORCH77 Orchard, R.A., "A New Methodology for Computer System Data Gathering", ACM SIGMETRICS Performance Evaluation Review, Vol. 6, No. 4, Fall 1977, pp. 27-41.

The author proposes that unbiased sampling techniques may produce better data at lower cost than the more common data gathering techniques currently in use. All sampling is done in a strictly boolean fashion and is collected via random sampling. Given a desired confidence interval, the number of samples required can be calculated.

- RAIC64 Raichelson, E., Collins, G., "A Method for Comparing the Internal Operating Speeds of Computers", Communication of the ACM, Vol. 7, No. 5, May 1964, pp.309-310.

The paper discusses using weighted instruction timings to derive overall machine timings figures that can be used for meaningful comparisons. Rather than just comparing memory access time or the time required for an add instruction, the real operation frequencies should be determined. The value used should be the weighted average based on the actual instruction timing and its relative frequency.

- RITA75 Ritacco, J., "Evolution of a Synthetic Jobstream Model", Share Inc. Computer Measurement and Evaluation - Selected Papers from the Share Project, Vol. 3, 1975, pp. 294-300.

This paper describes the evolution and the use of the tools as described by Klein (KLEI75). A batch jobstream has been modelled successfully such that the demands of the batch jobstream differ from the demands of the production jobstream, on which it was based, by about 10 percent. Particular problems experienced and the corresponding solutions are presented.

- ROZW73 Rozwadowski, R.T., "A Measure for the Quantity of Computation", Proc. First Annual SIGME Symposium on Measurement and Evaluation, February 1973, pp. 100-111.

Workload is defined in terms of the mechanical work. If energy, or work, is applied to some data, order is attained by the data. The energy is absorbed and traded for less disorder.

- SALT70 Saltzer, J.H., Gintell, J.W., "The Instrumentation of MULTICS", Communications of the ACM, Vol. 13, No. 8, August 1970, pp. 495-500.

The collection of tools to aid in measuring MULTICS is discussed. Among these tools are two types of benchmarking facilities. One is a program for a PDP-8 which, via telephone lines to the main system, simulates between one and twelve terminal users executing a given terminal script. There also exists an 'internal user' benchmark facility where the users are defined internally to the system with minor operating system modifications.

- SCHA75 Schatzoff, M., Tillman, C.C., "Design of Experiments in Simulator Validation", IBM Journal of Research and Development, Vol. 19, No. 3, May 1975, pp. 252-262.

The simulator studied was designed for testing new software algorithms for CP-67. Trace data is reduced and provided as input to the simulator. Experimental design considerations from the statistics point of view are discussed and used in the calibration and validation of the simulator.

- SCHW72 Schwemm, R.E., "Experience Gained in the Development and Use of TSS", AFIPS Proc. SJCC, 1972, pp. 559-569.

The experiences with TSS from 1965 through 1972 are presented. In particular, the experience gained with regard to system structure, system performance analysis, software development tools, and the management of software development are discussed. In the area of system performance, external tools, such as benchmarks, and internal tools, such as measurement tools, are discussed.

- SHER72 Sherman, S., Baskett, F. III, Browne, J.C., "Trace Driven Modelling and Analysis of CPU Scheduling in a Multiprogramming System", Communications of the ACM, Vol. 15, No. 12, December 1972, pp. 1063-1069.

A study done on the performance of various scheduling algorithms is discussed. A simulation model, driven by trace data collected by an event driven software probe, is used. The use of trace data as input, in addition to being low level, allows the use of actual data.

SHET74 Shetler, A.C., "Controlled Testing for Computer Performance Evaluation", AFIP Proc. NCC, 1974, pp. 693-699.

The author is concerned that performance studies must be done in a controlled environment for maximum effectiveness. In particular, the inputs and operating conditions must be controlled to assist in verifying execution characteristics. He considers batch jobstreams selected from the real workload in addition to synthetic batch streams and terminal scripts.

SHOP70 Shope, W.L., Kashmarack, K.L., Inghram, J.W., Decker, W.F., "System Performance Study", Proc. SHARE XXXIV, Vol. 1, March 1970, pp. 439-530.

The paper describes a study undertaken at the University of Iowa to determine how they could best service their users. Benchmarking and system performance measurement are used. The benchmark stream is selected at random from the real workload.

SREE74 Sreenivasan, K., Kleiman, A.J., "On the Construction of a Representative Synthetic Workload", Communications of the ACM, Vol. 17, No. 3, March 1974, pp. 127-133.

The paper is concerned with constructing a synthetic jobstream which is representative of the real workload statistics for some period. The statistics are collected in the form of resource demands. The synthetic job is based on that presented by Bucholz (BUCH69). The joint probability density of the drive workload is made to match that of the real workload. Calibration experiments are used to determine the relationship between the parameters of the synthetic program and the workload characteristics.

STRA72 Strauss, J.C., "A Benchmark Study", AFIPS Proc. FJCC, 1972, pp. 1225-1233.

A study done at the University of Washington to aid in a purchase decision is presented. The concern was to develop a fairly representative workload that would run on a wide variety of machines. This workload is used to determine the throughput of similarly priced machines. The primarily batch workload is composed of Fortran, Cobol, and Watfiv jobs.

- SVOB76 Svobodova, L., Computer Performance Measurement and Evaluation Methods: Analysis and Applications, Elsevier, New York, 1976.

The book presents comprehensive information on the current state of performance evaluation and measurement. Simulation and analytic modeling tools are discussed as well as workload modelling. There is also considerable material on measurement hardware and software. It has a comprehensive bibliography.

- SYMS74 Syms, G.H., "Benchmarked Comparisons of Terminal Support Systems for IBM 360 Computers", ACM SIGMETRICS Performance Evaluation Review, Vol. 2, No. 3, September 1974, pp. 6-34.

The paper describes a study ,using the IBM 360/67, to evaluate the performance of MTS, ISS/360, and CP/67 as time sharing systems, and TSS, MTS, and OS/MVT as batch systems. The systems are tested under varying loads with concern for the external performance, as seen by the user, as well as the internal performance. The loads are imposed via batch and terminal benchmarks using typical jobs, and via synthetic jobs developed to test performance under controlled resource demands.

- TOTA67 Totaro, J.B., "Real Time Processing Power: A Standardized Evaluation", Computers and Automation, Vol. 17, No. 4, April 1967, pp. 16-19.

The selection of random access storage devices is considered by means of a benchmark performing a typical application of an online inventory control application.

- WALD73 Waldbaum, G., "Evaluating Computing System Changes by Means of Regression Models", Proc. First Annual SIGME Symposium on Measurement and Evaluation,

February 1973, pp. 308-320.

The evolution of a regression model is used to study the effects of system modifications. Running APL on an IBM 360/91 under OS/MVT, he considers system response time as the dependent variable and usage of system resources and various system parameters as the independent variables. The cumulative density function of the dependent variable is considered rather than a simple value. A simple linear regression is proposed and then a quadratic regression. Stepwise regression is used to reduce the number of terms.

WIND73 Winder, R.O., "A Data Base for Computer Performance Evaluation", Computer, Vol. 6, No. 3, March 1973, pp. 25-29.

A technique used at R.C.A. to aid in the design of cache memories is described. A trace program collects information on the instructions being executed and an analysis program produces data on program behaviour.

WOOD71 Wood, D.C., Forman, E.H., "Throughput Measurements Using a Synthetic Job Stream", AFIPS Proc. FJCC, 1971, pp.51-55.

Experience gained in describing the characteristics of a workload and generating a synthetic job stream based on these characteristics is discussed. Specifications of the individual jobs are determined so that they match the workload. Synthetic jobs are then generated with these specifications. The synthetic job is based on that described by Bucholz (BUCH69). Comparable results occur from the actual jobstream and corresponding synthetic stream.

Appendix A - Experiment Schedule

- Pilot Study I - March and April, 1978
- Pilot Study II - January, 1979
- Pilot Study III - March and April, 1979
- Final Study - May and June, 1979

Appendix B - Clustering Results Data

The subsequent Figures and Tables are used as a reference in Section 7.1.3. A complete description of the Figures can be found there.

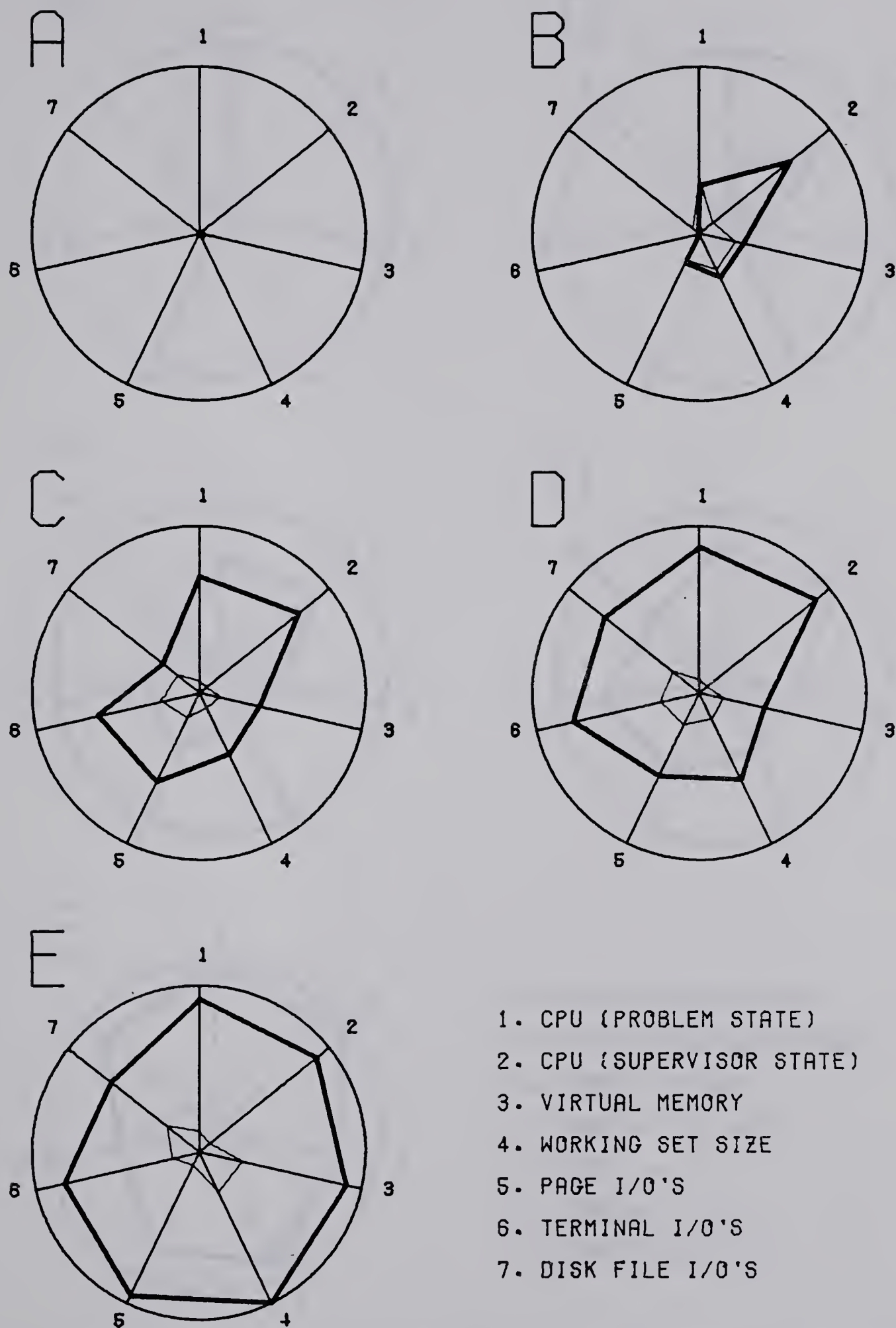


FIGURE 11 - Task Groups - May 24, 10 a.m. - Segment 1

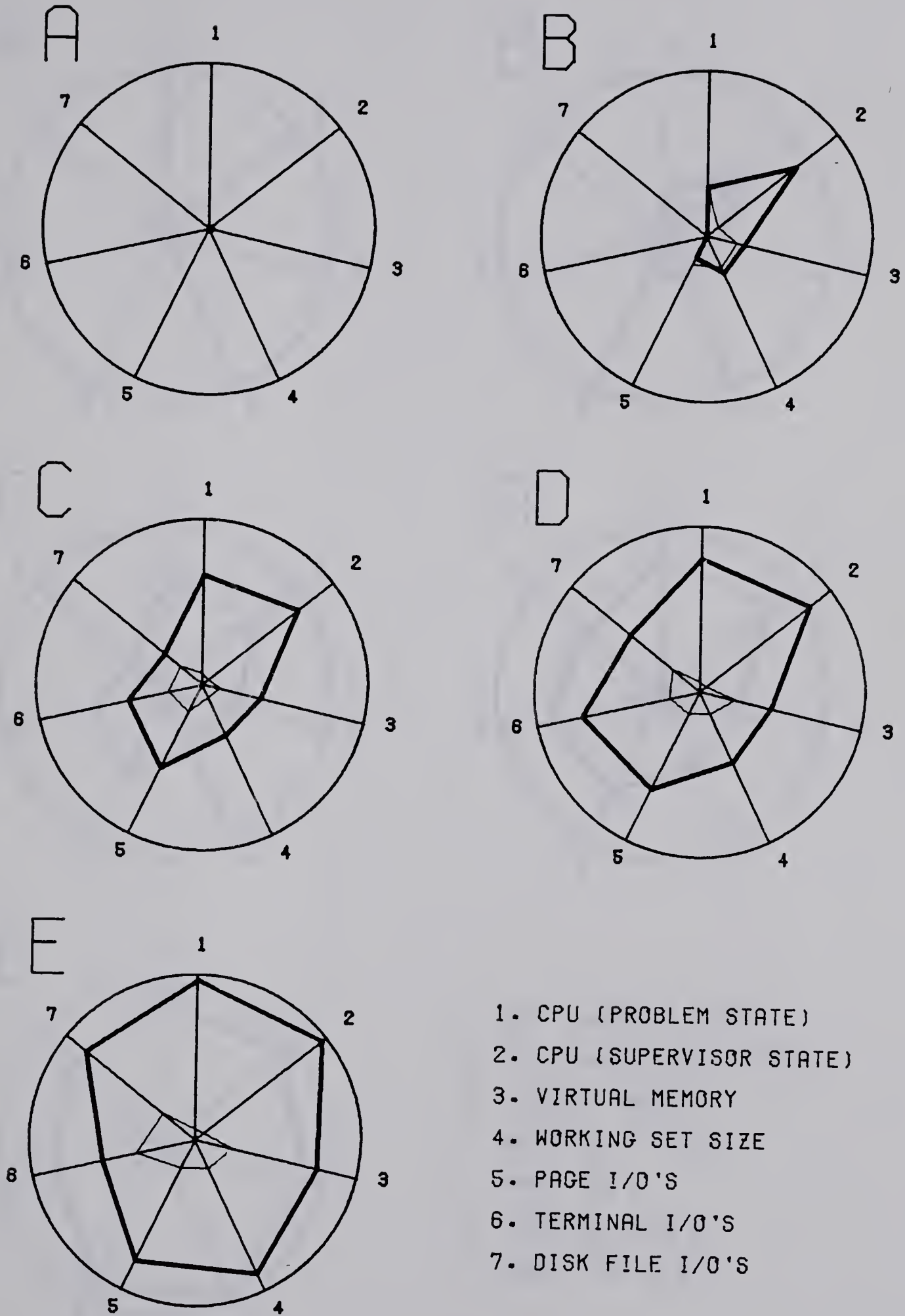


FIGURE 12 - Task Groups - May 24, 10 a.m. - Segment 2

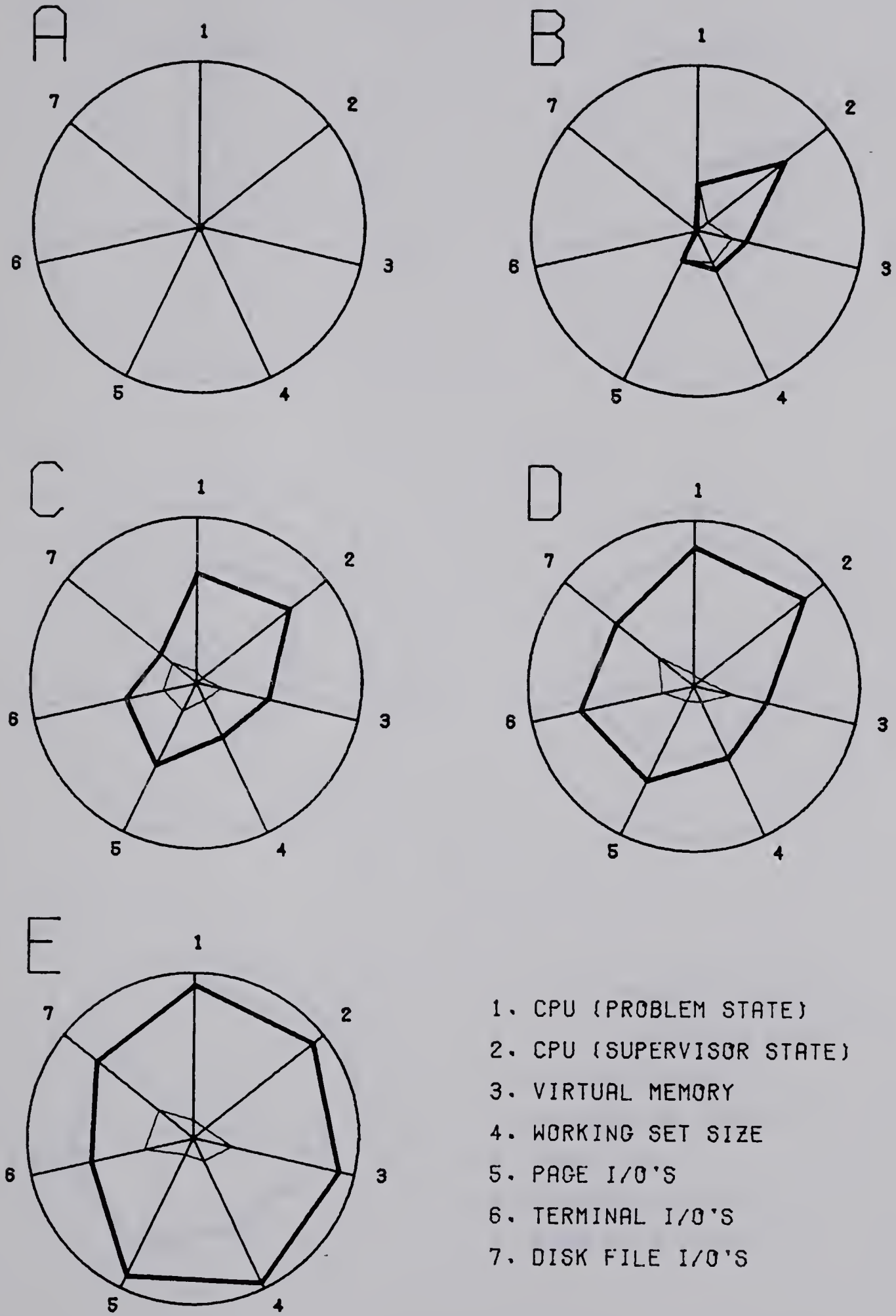


FIGURE 13 - Task Groups - May 24, 10 a.m. - Segment 3

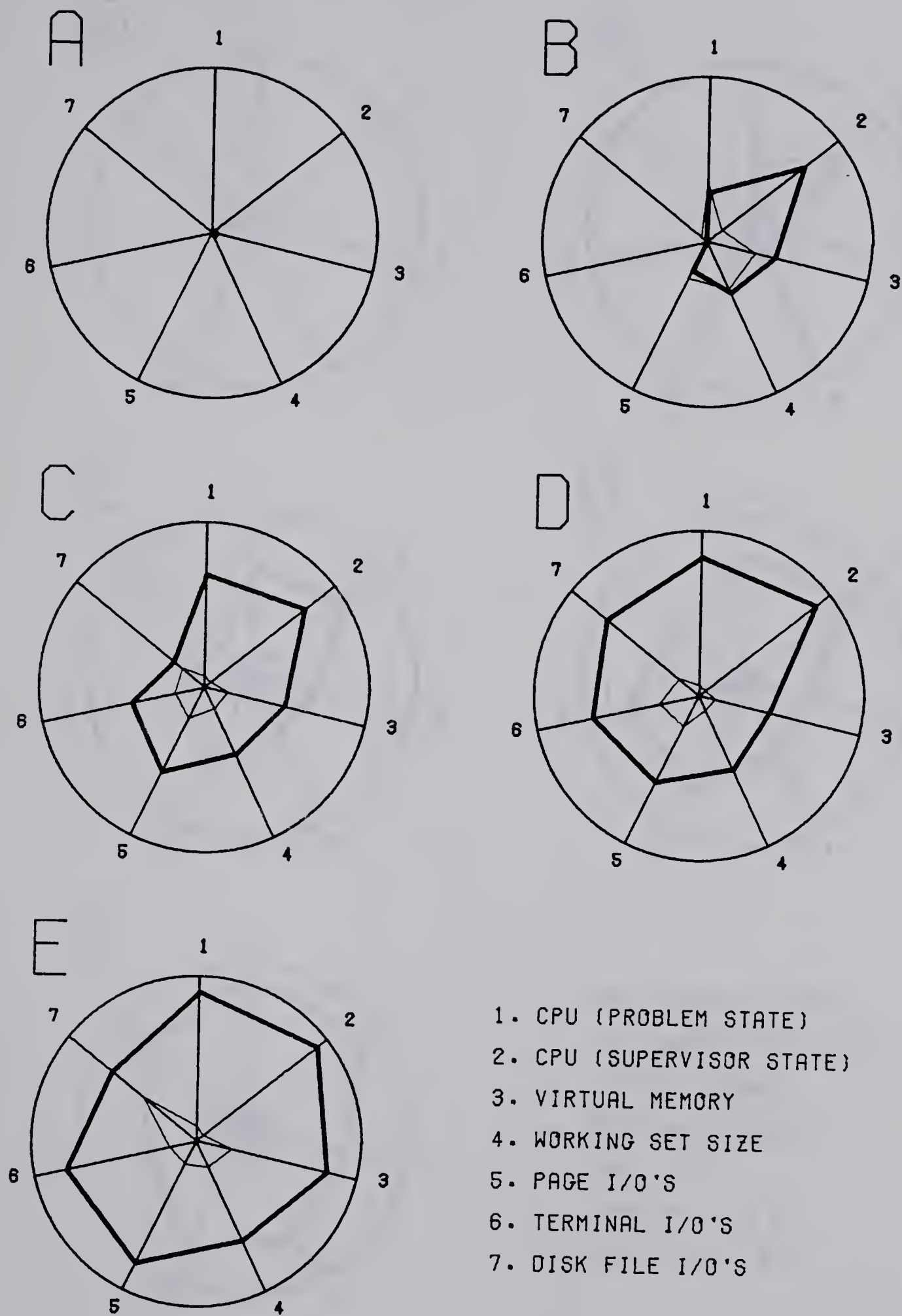


FIGURE 14 - Task Groups - May 24, 10 a.m. - Segment 10

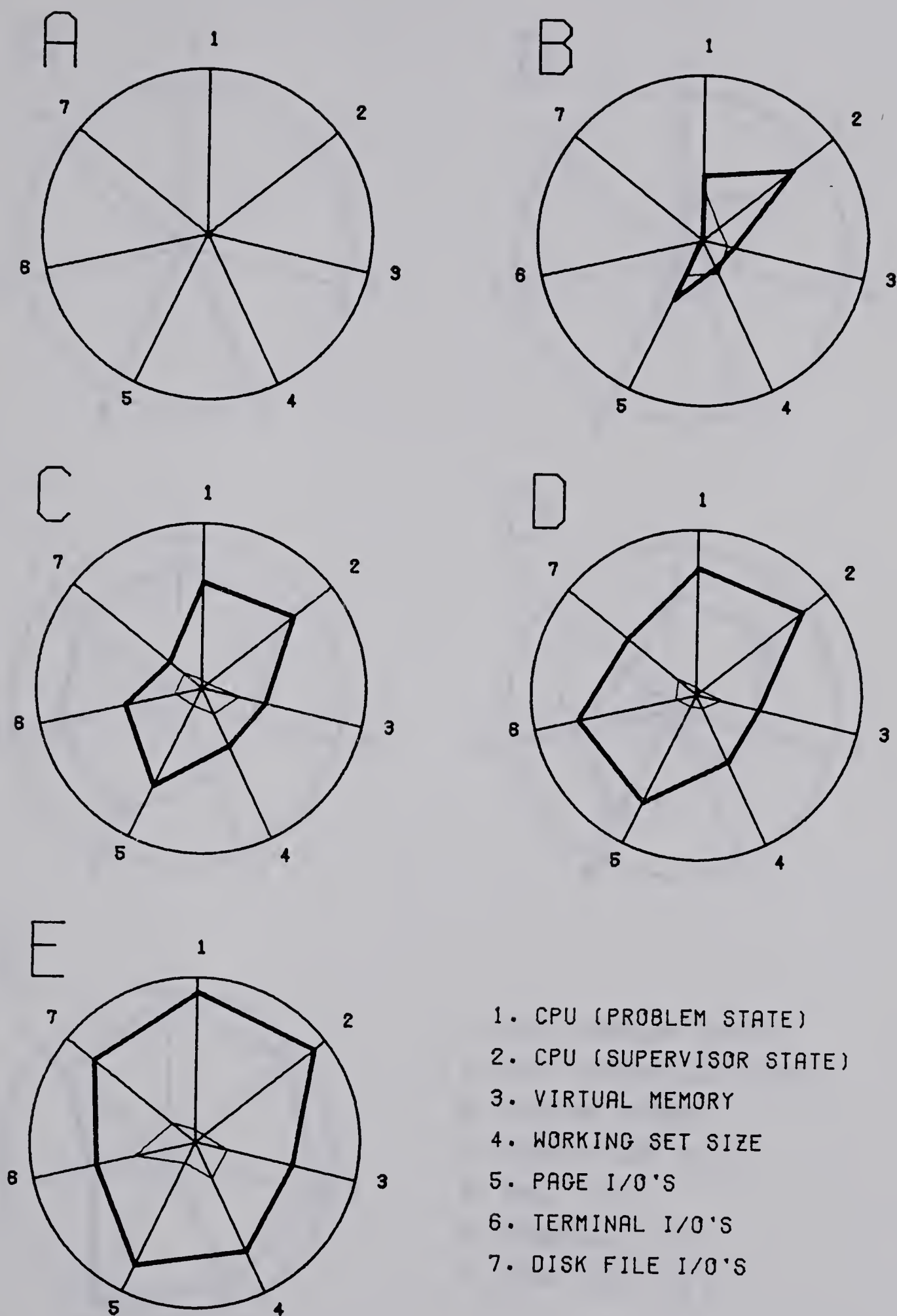


FIGURE 15 - Task Groups - May 24, 10 a.m. - Segment 20

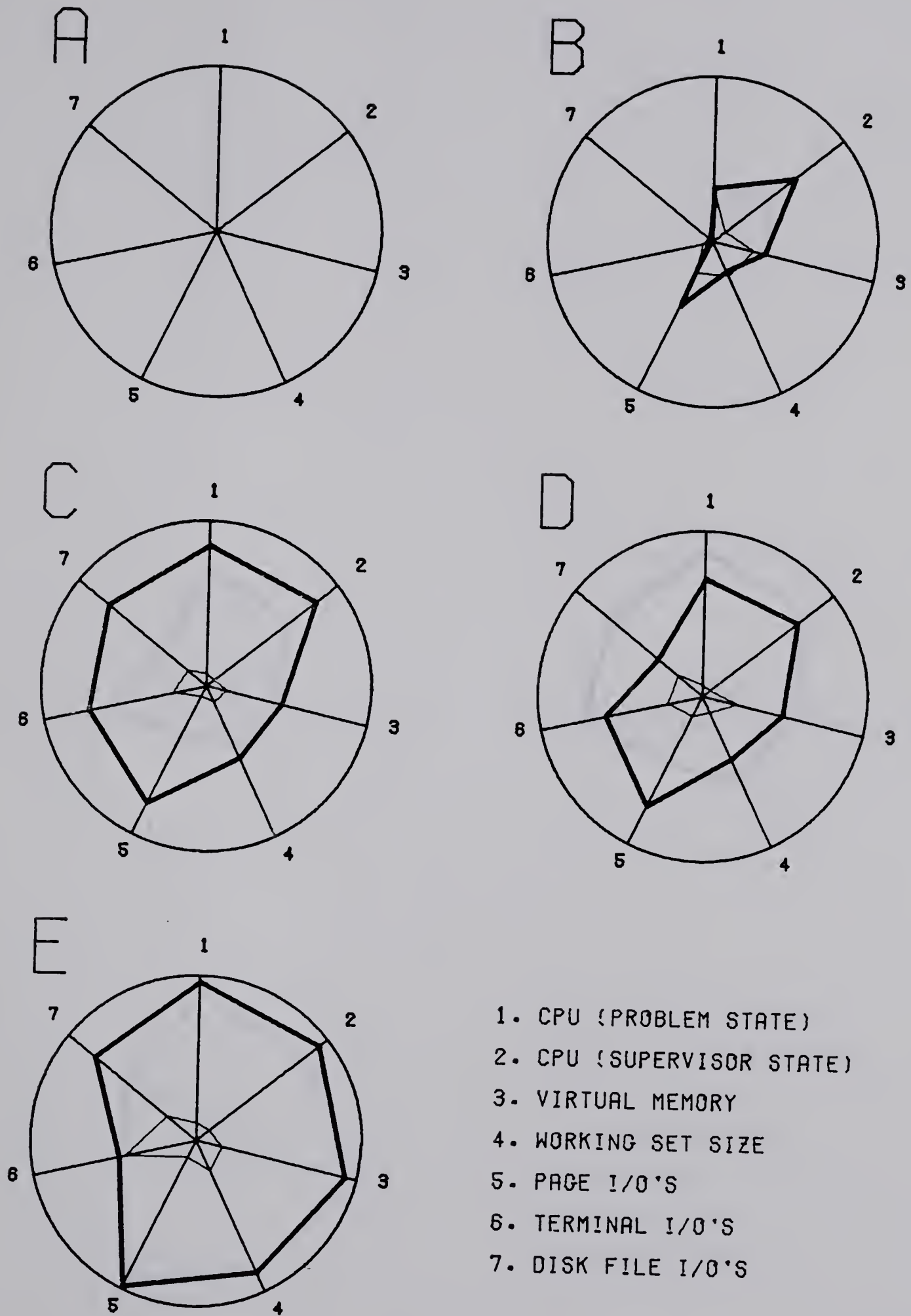


FIGURE 16 - Task Groups - June 7, 2 p.m. - Segment 1

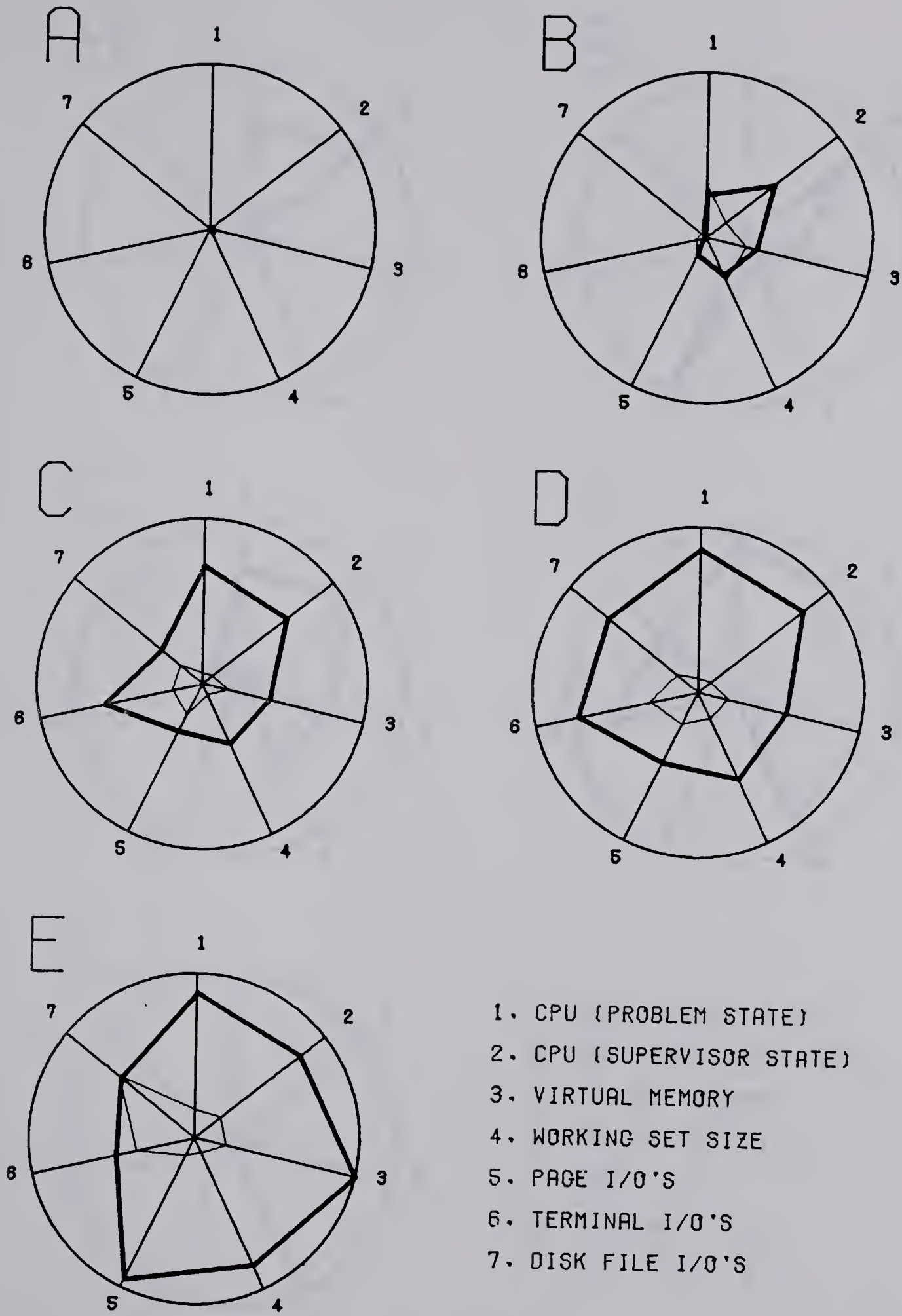


FIGURE 17 - Task Groups - June 22, noon - Segment 1

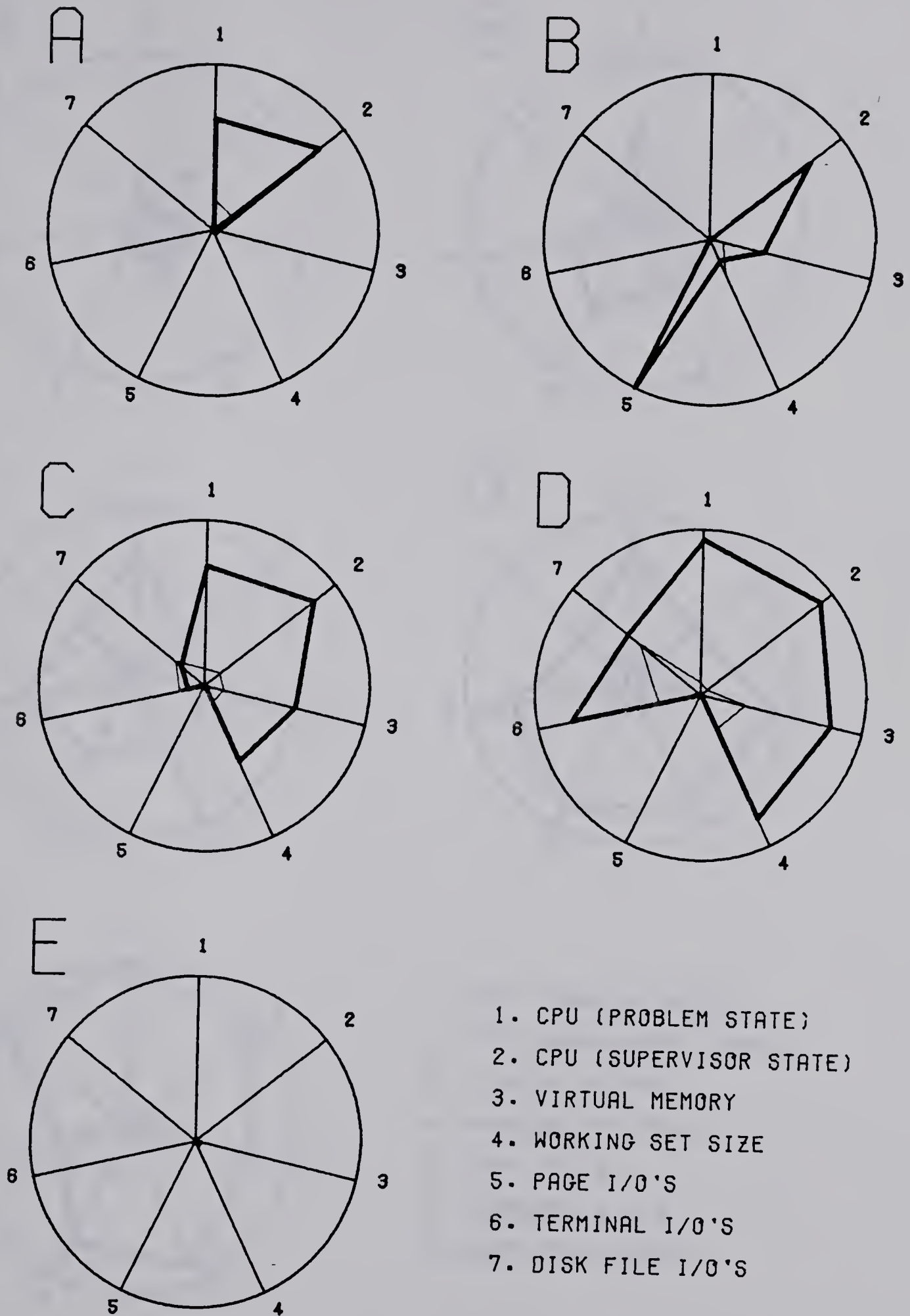


FIGURE 18 - Task Groups - June 27, 1 a.m. - Segment 1

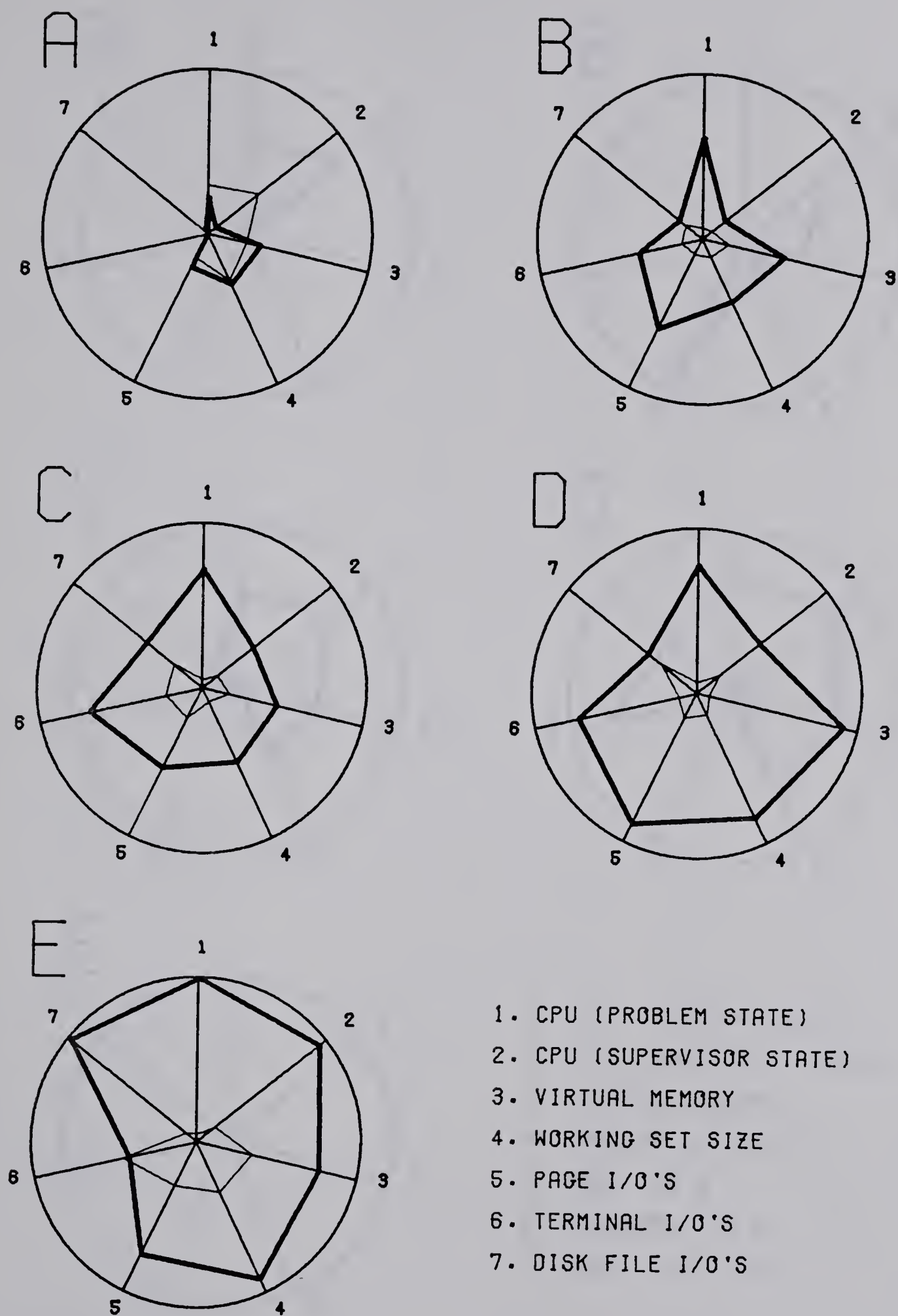


FIGURE 19 - Task Groups - June 27, 2 p.m. - Segment 1

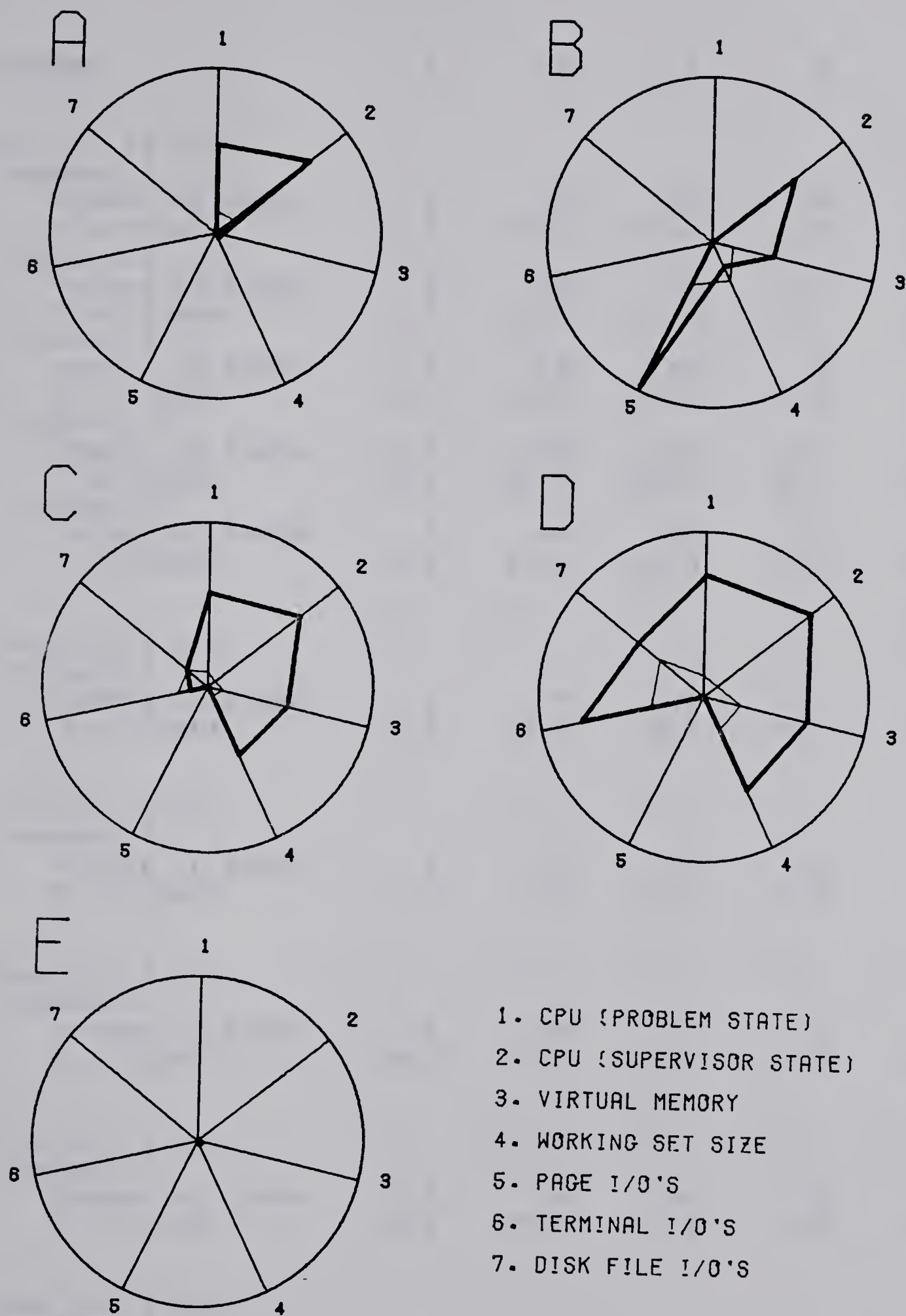


FIGURE 20 - Task Groups - June 28, 8 p.m. - Segment 1

Cluster	A	B	C	D	E
May 24, 10 a.m.					
Segment 1					
Number of Tasks	7	45	68	24	11
% of Tasks	4.5	29.0	43.8	15.4	7.0
Segment 2					
Number of Tasks	5	46	43	41	15
% of Tasks	3.3	30.6	28.6	27.3	10.0
Segment 3					
Number of Tasks	12	48	46	32	14
% of Tasks	7.8	31.5	30.2	21.0	9.2
Segment 10					
Number of Tasks	10	56	37	41	25
% of Tasks	5.9	33.1	21.8	24.2	14.7
Segment 20					
Number of Tasks	7	40	74	73	36
% of Tasks	3.0	17.3	32.1	31.7	15.6
June 07, 2 p.m.					
Segment 1					
Number of Tasks	9	61	50	107	24
% of Tasks	3.5	24.3	19.9	42.6	9.5
June 22, noon					
Segment 1					
Number of Tasks	3	62	48	32	5
% of Tasks	2.0	41.3	32.0	21.3	3.3
June 27, 1 a.m.					
Segment 1					
Number of Tasks	12	15	11	6	5
% of Tasks	24.4	30.6	22.4	12.2	10.2
June 27, 2 p.m.					
Segment 1					
Number of Tasks	70	39	66	15	10
% of Tasks	35.0	19.5	33.0	7.5	5.0
June 28, 8 p.m.					
Segment 1					
Number of Tasks	12	18	12	16	6
% of Tasks	18.7	28.1	18.7	25.0	9.3

TABLE 4 - Total Number of Tasks by Segment and Cluster

Cluster
(all values are
MTS - non-MTS)

A B C D E

May 24, 10 a.m.

Segment 1					
Minimum	0-7	32-12	53-0	13-0	8-0
Maximum	0-7	33-12	59-0	19-0	10-0
Average	0-7	33-12	55-0	15-0	9-0
Segment 2					
Minimum	0-5	32-14	40-0	27-0	8-0
Maximum	0-5	32-14	43-0	34-0	14-0
Average	0-5	32-14	42-0	30-0	12-0
Segment 3					
Minimum	0-7	34-12	41-0	26-0	9-0
Maximum	1-7	35-12	46-0	28-0	12-0
Average	0-7	34-12	42-0	27-0	10-0
Segment 10					
Minimum	0-8	41-10	33-0	31-0	21-0
Maximum	1-8	43-10	35-0	36-0	24-0
Average	0-8	42-10	34-0	34-0	23-0
Segment 20					
Minimum	0-5	21-14	56-0	45-0	20-0
Maximum	1-5	26-14	62-0	61-0	26-0
Average	0-5	22-14	60-0	58-0	22-0

TABLE 5 - Active Tasks by Segment Cluster - Part I

Cluster (All values are MTS - non-MTS)	A	B	C	D	E
June 07, 2 p.m.					
Segment 1					
Minimum	0-3	42-15	28-0	76-0	13-0
Maximum	1-3	44-16	35-0	86-0	15-0
Average	0-3	43-15	29-0	80-0	14-0
June 22, noon					
Segment 1					
Minimum	0-3	47-14	41-0	26-0	4-0
Maximum	0-3	48-14	46-0	30-0	4-0
Average	0-3	47-14	45-0	27-0	4-0
June 27, 1 a.m.					
Segment 1					
Minimum	0-10	15-0	11-0	6-0	0-5
Maximum	0-11	15-0	11-0	6-0	0-5
Average	0-11	15-0	11-0	6-0	0-5
June 27, 2 p.m.					
Segment 1					
Minimum	48-18	38-0	54-0	12-0	3-0
Maximum	50-18	39-0	58-0	14-0	5-0
Average	49-18	39-0	55-0	13-0	4-0
June 28, 8 p.m.					
Segment 1					
Minimum	0-12	18-0	10-0	14-0	0-6
Maximum	0-12	18-0	11-0	15-0	0-6
Average	0-12	18-0	11-0	15-0	0-6

TABLE 6 - Active Tasks by Segment and Cluster - Part II

Cluster	A	B	C	D	E	F
May 24, 10 a.m. - Segment 1						
CPU (problem) ¹	0	0	.2	3.8	7.2	N/A
CPU (supervisor) ¹	0	.1	.4	2.7	3.0	N/A
Virtual Memory ²	.7	10.4	16.5	18.8	100.0	N/A
Working Set Size ²	.5	6.0	9.3	15.2	42.4	N/A
Page I/O's ³	0	13.6	131.5	107.9	674.4	N/A
Terminal I/O's ³	0	0	160.9	328.7	418.4	N/A
Disk File I/O's ³	0	.6	83.3	3372.5	2205.3	N/A
May 24, 10 a.m. - Segment 2						
CPU (problem) ¹	0	0	.1	1.6	19.5	N/A
CPU (supervisor) ¹	0	.1	.3	1.4	9.0	N/A
Virtual Memory ²	.5	9.0	17.2	23.9	70.7	N/A
Working Set Size ²	.3	5.5	8.4	13.4	39.5	N/A
Page I/O's ³	0	9.8	130.4	210.4	431.8	N/A
Terminal I/O's ³	0	0	103.3	473.9	202.5	N/A
Disk File I/O's ³	0	0	95.3	795.8	9951.5	N/A
May 24, 10 a.m. - Segment 3						
CPU (problem) ¹	0	0	.1	2.4	8.8	N/A
CPU (supervisor) ¹	0	.1	.2	1.9	5.5	N/A
Virtual Memory ²	.4	10.3	18.8	19.0	75.1	N/A
Working Set Size ²	.3	5.4	8.2	12.1	39.0	N/A
Page I/O's ³	0	15.5	125.2	192.4	720.1	N/A
Terminal I/O's ³	0	.1	95.7	421.6	291.2	N/A
Disk File I/O's ³	0	.4	79.3	1104.3	3658.1	N/A
May 24, 10 a.m. - Segment 10						
CPU (problem) ¹	0	0	.1	1.2	3.2	N/A
CPU (supervisor) ¹	0	.1	.2	1.2	2.0	N/A
Virtual Memory ²	.2	13.9	18.2	14.8	44.2	N/A
Working Set Size ²	.3	6.9	10.1	11.4	17.9	N/A
Page I/O's ³	0	12.8	105.4	112.1	316.9	N/A
Terminal I/O's ³	0	0	88.5	271.5	580.1	N/A
Disk File I/O's ³	0	.5	43.4	1566.5	1047.9	N/A

¹ in [(seconds of CPU time)/(second of real time) * 1000]

² in pages

³ in [(I/O's)/(second of real time) * 1000]

TABLE 7 - Actual Unscaled Consumption Rates - Part I

Cluster	A	B	C	D	E	F
May 24, 10 a.m. - Segment 20						
CPU (problem) ¹	0	0	.1	1.2	11.2	N/A
CPU (supervisor) ¹	0	.1	.2	.9	4.2	N/A
Virtual Memory ²	.1	7.8	22.4	21.6	49.7	N/A
Working Set Size ²	.1	3.6	9.4	11.5	24.4	N/A
Page I/O's ³	0	73.3	314.7	457.0	763.1	N/A
Terminal I/O's ³	0	.3	92.2	352.1	192.5	N/A
Disk File I/O's ³	0	.4	73.0	900.8	7878.1	N/A

June 7, 2 p.m. - Segment 1						
CPU (problem) ¹	0	0	3.5	.4	18.1	22.9
CPU (supervisor) ¹	0	.1	2.3	.4	11.7	59.6
Virtual Memory ²	.1	11.8	20.9	23.2	84.6	1095.9
Working Set Size ²	0	4.0	12.7	10.4	34.6	26.9
Page I/O's ³	0	76.6	490.8	390.3	1269.2	1450.9
Terminal I/O's ³	0	1.2	336.6	180.7	93.1	7.9
Disk File I/O's ³	0	1.0	4396.8	152.7	5606.7	.9

June 22, noon - Segment 1						
CPU (problem) ¹	0	0	.2	2.3	2.9	N/A
CPU (supervisor) ¹	0	.1	.3	1.5	1.6	N/A
Virtual Memory ²	.6	11.8	18.4	29.3	119.5	N/A
Working Set Size ²	.6	6.2	11.5	20.4	40.7	N/A
Page I/O's ³	0	6.5	28.8	65.6	537.9	N/A
Terminal I/O's ³	0	.8	198.5	394.4	105.3	N/A
Disk File I/O's ³	0	.5	130.6	3233.3	1175.2	N/A

June 27, 1 a.m. - Segment 1						
CPU (problem) ¹	0	0	0	.7	0	N/A
CPU (supervisor) ¹	.1	0	.1	.3	0	N/A
Virtual Memory ²	.5	8.9	19.2	36.4	.3	N/A
Working Set Size ²	.4	3.0	15.4	35.8	.3	N/A
Page I/O's ³	0	6.0	0	0	0	N/A
Terminal I/O's ³	0	0	5.8	323.7	0	N/A
Disk File I/O's ³	0	0	20.8	312.4	0	N/A

¹ in [(seconds of CPU time) / (second of real time) * 1000]

² in pages

³ in [(I/O's) / (second of real time) * 1000]

TABLE 8 - Actual Unscaled Consumption Rates - Part II

Cluster	A	B	C	D	E	F
June 27, 2 p.m. - Segment 1						
CPU (problem) ¹	0	.1	.5	1.3	38.6	13.6
CPU (supervisor) ¹	0	.1	.6	1.0	17.9	24.9
Virtual Memory ²	10.8	21.3	18.4	66.9	45.1	8.9
Working Set Size ²	6.9	9.2	11.7	26.3	31.2	6.9
Page I/O's ³	17.6	127.9	93.0	438.9	247.0	0
Terminal I/O's ³	0	62.6	320.9	391.6	71.7	0
Disk File I/O's ³	.7	38.2	438.9	266.1	57703.8	998.9

June 28, 8 p.m. - Segment 1						
CPU (problem) ¹	0	0	.1	1.4	0	N/A
CPU (supervisor) ¹	.2	0	.1	.8	0	N/A
Virtual Memory ²	.7	9.6	14.0	21.3	.4	N/A
Working Set Size ²	.6	3.4	12.3	20.2	.4	N/A
Page I/O's ³	0	5.8	0	0	0	N/A
Terminal I/O's ³	0	0	5.3	189.8	0	N/A
Disk File I/O's ³	0	0	30.2	712.9	0	N/A

¹ in [(seconds of CPU time) / (second of real time) * 1000]

² in pages

³ in [(I/O's) / (second of real time) * 1000]

TABLE 9 - Actual Unscaled Consumption Rates - Part III

B30257